REALISTIC IMAGE SYNTHESIS WITH LIGHT TRANSPORT

HUA BINH SON
Bachelor of Engineering

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE

2015

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Hua Binh Son
January 2015

Acknowledgements

I would like to express my sincere gratitude to Dr. Low Kok Lim for his continued guidance and support on every of my projects during the last six years. He brought me to the world of computer graphics and taught me progressive radiosity, my very first lesson about global illumination, which was later set to be the research theme for this thesis. Great thanks also go to Dr. Ng Tian Tsong for his advice and collaboration in the work in Chapter 7, and to Dr. Imari Sato for her kind guidance and collaboration in the work in Chapter 6. I also thank Prof. Tan Tiow Seng for guiding the G^3 lab students including me on how to commit to high standards in all of our work.

I would like to take this opportunity to thank my G^3 lab mates for accompanying me in this long journey. I thank Cao Thanh Tung for occasional discussions about trending technologies which keeps my working days less monotonic; Rahul Singhal for discussing about principles of life and work of a graduate student; Ramanpreet Singh Pahwa for collaborating on the depth camera calibration project; Cui Yingchao and Delia Sambotin for daring to experiment with my renderer and the interreflection reconstruction project; Liu Linlin, Le Nguyen Tuong Vu, Wang Lei, Li Ruoru, Ashwin Nanjappa, and Conrado Ruiz for their accompany in the years of this journey. Thanks also go to Le Duy Khanh, Le Ton Chanh, Ta Quang Trung, and my other friends for their help and encouragement.

Lastly, I would like to express my heartfelt gratitude to my family for their continuous and unconditional support.

Abstract

In interior and lighting design, 3D animation, and computer games, it is always demanded to produce visually pleasant content to users and audience. A key to achieve this goal is to render scenes in a physically correct manner and account for all types of light transport in the scenes, including direct and indirect illumination. Rendering from given scene data can be regarded as *forward light transport*.

In augmented reality, it is often required to render a scene that has real and virtual objects placed together. The real scene is often captured and scene information is extracted to provide input to rendering. For this task, light transport matrix can be used. *Inverse light transport* is the process of extracting scene information from a light transport matrix, e.g., geometry and materials. Understanding both forward and inverse light transport are therefore important to produce realistic images.

This thesis is a two-part study about light transport. The first part is dedicated to forward light transport, which focuses on global illumination and many-light rendering. First, a new importance sampling technique which is built upon virtual point light and the Metropolis-Hastings algorithm is presented. Second, an approach to reduce artifacts in many-light rendering is proposed. Our experiments show that our techniques can improve the effectiveness in many-light rendering by reducing noise and visual artifacts.

The second part of the thesis is a study about inverse light transport. First, an extension to compressive dual photography is presented to accelerate the demultiplexing of dual images, which is useful for preview for light transport capturing. Second, a new formulation to acquire geometry from radiometric data such as interreflections is presented. Our experiments with synthetic data show that depth and surface orientation can be reconstructed by solving a system of polynomials.

Contents

List of Figures viii

List of Tables xi

List of Algorithms xii

1 Introduction **1**

2 Fundamentals of realistic image synthesis **4**

2.1 Radiometry 4

2.1.1 Radiance 4

2.1.2 Invariance of radiance in homogeneous media 5

2.1.3 Solid angle 6

2.1.4 The rendering equation 7

2.1.5 The area integral 8

2.1.6 The path integral 9

2.2 Monte Carlo integration 10

2.2.1 Monte Carlo estimator 10

2.2.2 Solving the rendering equation with Monte Carlo estimators 12

2.3 Materials 14

2.3.1 The Lambertian model 15

2.3.2 Modified Phong model 16

2.3.3 Anisotropic Ward model 19

2.3.4 Perfect mirror 20

2.3.5 Glass 20

2.4 Geometry 22

2.4.1 Octree 22

2.4.2 Sampling basic shapes 23

2.5 Light 24

2.5.1 Spherical light 24

2.5.2 Rectangular light 25

3 Global illumination algorithms **27**

3.1 Direct illumination 27

3.1.1 Multiple importance sampling 28

3.2 Unidirectional path tracing 29

3.2.1 Path tracing 29

3.2.2 Light tracing 30

3.3 Bidirectional path tracing 31
3.3.1 State of the arts in path tracing 34
3.4 Photon mapping 35
3.5 Many-light rendering 36
3.5.1 Generating VPLs and VPSes 37
3.5.2 Gathering illumination from VPLs 37
3.5.3 Visibility query 39
3.5.4 Progressive many-light rendering 40
3.5.5 Bias in many-light rendering 40
3.5.6 Clustering of VPLs 41
3.5.7 Glossy surfaces 41
3.6 Interactive and real-time global illumination 42
3.7 Conclusions 44

4 Guided path tracing using virtual point lights 45
4.1 Related works 46
4.1.1 Many-light rendering 46
4.1.2 Importance sampling with VPLs 48
4.2 Our method 49
4.2.1 Estimating incoming radiance 50
4.2.2 Metropolis sampling 50
4.2.3 Estimating the total incoming radiance 53
4.2.4 Sampling the product of incoming radiance and BRDF 53
4.2.5 VPL clustering 54
4.3 Implementation details 54
4.4 Experimental results 55
4.5 Conclusions 59

5 Reducing artifacts in many-light rendering 60
5.1 Related works 62
5.2 Virtual point light 63
5.3 Our method 64
5.3.1 Generating the clamping map 64
5.3.2 Analyzing the clamping map 65
5.3.3 Generating extra VPLs 66
5.3.4 Implementation details 67
5.4 Experimental results 68
5.5 Conclusions 70

6 Direct and progressive reconstruction of dual photography images 71
6.1 Dual photography 71

6.2 Related works 72
6.3 Compressive dual photography 74
6.4 Direct and progressive reconstruction 75
6.4.1 Direct reconstruction 75
6.4.2 Progressive reconstruction 76
6.5 Implementation 77
6.6 Experiments 78
6.6.1 Running time analysis 79
6.7 More results 80
6.8 Discussion 81
6.9 Conclusions 81

7 Reconstruction of depth and normals from interreflections 83
7.1 Geometry from light transport 83
7.2 Related works 85
7.2.1 Conventional methods 85
7.2.2 Hybrid methods 86
7.2.3 Reconstruction in the presence of global illumination 86
7.3 Interreflections in light transport 88
7.4 Geometry reconstruction from interreflections 89
7.4.1 Polynomial equations from interreflections 89
7.4.2 Algorithm to recover location and orientation 90
7.4.3 Implementation 90
7.5 Experiments 91
7.6 Conclusions 92

8 Conclusions 93

References 94

A More implementation details 102
A.1 Probability density function 102
A.1.1 Changing variables in probability density function 102
A.1.2 Deriving cosine-weighted sampling formula 102
A.2 Form factor 103
A.3 Conversion between VPL and photon 104
A.3.1 Reflected radiance using photons 104
A.3.2 Reflected radiance using VPLs 104
A.3.3 From photon to VPL 105
A.3.4 From VPL to photon 105
A.4 Hemispherical mapping 106

List of Figures

2.1 From left to right: flux, radiosity, and radiance. 5

2.2 Solid angle. 7

2.3 Three-point light transport. 9

2.4 Sampling the Phong BRDF model. 17

2.5 Sampling the Ward BRDF model based on the half vector ω_h. 19

2.6 The modified Cornell box. 21

2.7 A 2D visualization of a quad-tree. Thickness of the border represents the level of a tree node. The thickest border represents the root. 23

2.8 Sampling spherical and rectangular light. 25

3.1 Sampling points on the light sources vs. sampling directions from the BSDF. Figure derived from [Gruenschloss et al. 2012] (see page 14). 28

3.2 Multiple importance sampling. Images are rendered with 64 samples. 29

3.3 Path tracing. 31

3.4 Direct illumination and global illumination. The second row is generated by path tracing. The Sibenik and Sponza scene are from [McGuire 2011]. 32

3.5 The modified Cornell box rendered by (a) light tracing and (b) path tracing. Note the smoother caustics with fewer samples in (a). 32

3.6 Different ways to generate a complete light path. 33

3.7 The Cornell box rendered by many-light rendering. 38

3.8 Complex scenes rendered by many-light rendering. The Kitchen scene is from [Hardy 2012], the Natural History and the Christmas scene from [Birn 2014]. 38

3.9 The gathering process with VPLs generated by tracing (a) light paths and (c)-(e) eye paths of length two. 42

4.1 An overview of our approach. We sample directions based on the distribution of incoming radiance estimated by virtual point lights. The main steps of our approach is as follows. (a) A set of VPLs is first generated. (b) Surface points visible to camera are generated and grouped into clusters based on their locations and orientations. The representatives of the clusters are used as cache points which store illumination from the VPLs and guide directional sampling. (c) The light transport from the VPLs to the cache points are computed. To support scalability, for each cache point, the VPLs are clustered adaptively by following LightSlice [Ou and Pellacini 2011]. (d) We can now sample directions based on incoming radiance estimated by the VPL clusters. At each cache point, we store a sample buffer and fill it with directions generated by the Metropolis algorithm. (e) In Monte Carlo path tracing, to sample at an arbitrary surface point, we query the nearest cache point and fetch a direction from its sample buffer. 46

4.2 Visualization of incoming radiance distributions at various points in the Cornell box scene, from left to right: (i) Incoming radiance as seen from the nearest cache point; (ii) The density map; (iii) Histogram from the Metropolis sampler; (iv) Ground truth incoming radiance seen from the gather point. 51

4.3 Absolute error plots of the example scenes. While Metropolis sampling does not always outperform BRDF sampling, combining both of the techniques using MIS gives far more accurate results. 56

4.4 The results of our tested scenes. Odd rows: results by Metropolis sampling, BRDF sampling, MIS, and by Vorba et al. [2014]. Even rows: error heat map of Metropolis sampling, BRDF sampling, MIS, and the ground truth. 58

5.1 Progressive rendering of the Kitchen scene [Hardy 2012]. Our method allows progressive rendering with less bright spots. 61

5.2 A clamping map from the Kitchen scene. 65

5.3 Extra VPLs are generated by sampling the cone subtended by a virtual sphere at the VPL that causes artifacts. 66

5.4 Progressive rendering of the Conference scene [McGuire 2011]. Similarly, our method allows progressive rendering with less bright spots. 69

5.5 The error plot of our tested scenes. The horizontal axis represents the total number of VPLs (in thousands). The vertical axis shows the absolute difference with the ground truth generated by path tracing. 70

6.1 Dual photography. (a) Camera view. (b) Dual image directly reconstructed from 16000 samples, which is not practical. (c) Dual image progressively reconstructed from only 1000 samples using our method with 64 basis dual images. (d) Dual image reconstructed with settings as in (c) but from 1500 samples. Haar wavelet is used for the reconstruction. 73

6.2 Comparison between direct and progressive reconstruction. Dual image (a), (b), and (c) are from direct reconstruction. Dual image (d) and (e) are from progressive reconstruction with 64 basis dual images. (f) Ground truth is generated from light transport from 16000 samples by inverting the circulant measurement matrix. Daubechies-8 wavelet is used for the reconstruction. 76
6.3 Progressive results of the dual image in Figure 6.1(d) by accumulating those reconstructed basis dual images. Our projector-camera setup to acquire light transport is shown in the diagram. 78
6.4 Relighting of the dual image in Figure 6.2(e). 80
6.5 Dual photography. (a) Camera view and generated images for capturing light transport. The projector is on the right of the box. (b) Dual image and the progressive reconstruction (floodlit lighting) from 4000 samples using our method with 256 basis dual images. Haar wavelet is used for the reconstruction. Image size is 256×256. 81

7.1 (a) Synthetic light transport using radiosity. (b) Reconstructed points from exact data by form factor formula. (c) Reconstructed points from data by radiosity renderer. 84
7.2 Reconstruction results with noise variance 10^{-2} and 10^{-1} added to input images. 91

List of Tables

4.1 Statistics of our scenes rendered using MIS. 59

List of Algorithms

4.1 The Metropolis algorithm to sample new directions and fill the sample buffer. The current direction in the Markov chain is ω. 52

Chapter 1

Introduction

Physically based rendering is an important advance in computer graphics in the last three decades. The reproduction of appearance of computer-synthesized objects has been increasingly more realistic. Such advances have been applied to several applications including movie and 3D animation production, interior and lighting design, and computer games which often require to produce visually pleasing content to audience. One of the keys to render a scene physically correct is to account for all types of light transport in the scene.

Essentially, there are two types of light transport: light from emitter to surface and from surface to surface. Illumination at a surface due to emitter-surface transport is called *direct illumination*. Similary, illumination due to surface-surface transport is called *indirect illumination*. Illumination that contains both types of transport is called *global illumination*. Direct illumination is the easiest to compute but only produces a moderate level of realism. It had been used in the early days of 3D animation production due to the limit of computation power. Indirect illumination is more complex to estimate, but it adds a great level of realism on top of direct illumination to the image rendition. Nowadays, with the advance of processors and graphics processors, global illumination has been necessary to render in both in movie, 3D animation, and game production. In legacy rendering pipelines, global illumination is simulated by lighting artists who might try to place several lights in a scene so that the final render has a realistic look. The next decade would see physically correct global illumination to become a part of the rendering pipeline, which would greatly improve realism and reduce the time necessary for lighting edit to simulate global illumination. The process of computing global illumination for a synthetic scene can be regarded as an implicit construction of the light transport, which represents the total amount of energy from light emitters to sensors after bouncing at scene surfaces. This can be regarded as *forward light transport*.

In parallel to rendering from synthetic data, there exists a class of rendering techniques that take images as input. Such image-based rendering methods work by manipulating images captured in a real world scene. This can also be regarded as an explicit construction of the light transport of a real world scene by many images. Image data in a light transport can be recombined to generate novel views of the real world scene; it can also be used to infer geometry, material, and light to create a virtual scene that accurately matches the real world scene. In the latter case, the virtual scene can then be the input to a physically based rendering algorithm in forward light transport. The analysis of the light transport in the latter case can be regarded as *inverse light transport*.

For example, an important step in movie production is to enable actors and real objects to interact with virtual objects synthesized by a computer. To achieve realism, it is necessary to simulate virtual objects to make them appear as if they were there in the scene. Their appearance needs to match the illumination from its environment and they need to interact correctly with other objects. In such cases, lighting, geometry, materials, and textures of real objects and the environment can be captured. Such data can be used in the post processing to synthesize the appearance and behavior of virtual objects. In this case, understanding in both forward and inverse light transport are important to create realistic images.

While forward light transport has been receiving great attentions from the computer graphics community, inverse light transport has been less mainstream due to the lengthy time to capture and reconstruct the light transport from a large volume of data. In computer vision, analysis tasks have been done massively on single-shot images or image sets and databases from the Internet. Very few works have focused on extracting scene information from a light transport captured by tens of thousands of images.

This thesis is a study about light transport. It has two parts that target forward and inverse light transport, respectively. The first part is dedicated to many-light rendering, a physically based forward rendering approach that is closely related to explicit construction of light transport in practice. Two problems in many-light rendering, importance sampling using virtual point lights, and artifact removal in many-light rendering are addressed. The second part is a study of inverse light transport. Two problems in light transport acquisition and analysis are addressed. Exploring both forward and inverse light transport is important to make a step further towards a more ambitious goal: to bring more accurate indirect illumination models in physically based rendering to inverse light transport, and to capture light transport in a real scene for guiding physically based rendering.

The contributions of this thesis are:

- A robust approach to importance sample the incoming radiance field for Monte Carlo path tracing which utilizes virtual light distribution from many-light rendering and clustering.
- An approach to reduce sharp artifacts in many-light rendering.
- An efficient approach to preview dual photography images, which facilitates the process of high-dimensional light transport acquisition.
- An algorithm to extract geometry from interreflection in a light transport.

This thesis is organized into two parts, the first part (Chapter 2, 3, 4, 5) for forward light transport, and the second part (Chapter 6, 7) for inverse light transport. In the first part, Chapter 2 introduces the core concepts in realistic image synthesis: radiometry, rendering equations, and Monte Carlo integration. Models for material, geometry, and light, which are the three must-have data sets of a scene in order to form an image, are discussed. Chapter 3 discusses the core algorithms and recent advances in global illumination: path tracing, bidirectional path tracing, photon mapping, and many-light rendering. Chapter 4 and Chapter 5 explore two important problems in many-light rendering: importance sampling using virtual point lights, and artifact removal. In the second part, Chapter 6 presents the fundamentals of light transport acquisition together with dual photography, an approach to acquire high-dimensional light transport. A fast and progressive solution to synthesize dual photography images is presented. Chapter 7 further investigates inverse light transport and presents an approach to reconstruct geometry from interreflection. Finally, Chapter 8 provides conclusions to this thesis.

Chapter 2

Fundamentals of realistic image synthesis

This chapter presents fundamental principles in realistic image synthesis. First, we define the common terms in radiometry such as flux, irradiance, radiosity, radiance, solid angles, and then present the rendering equation in solid-angle form. We then discuss each component in the rendering equation in details and present two other forms of the rendering equation, the area formulation and the path formulation. Second, we discuss about material system and the bidirectional reflectance distribution function (BRDF) which defines the look-and-feel of scene surfaces. Third, we discuss Monte Carlo integration, a stochastic approach that is widely used to solve the rendering equation. We then discuss importance sampling techniques, from the well-known cosine-weighted sampling to sampling techniques for commonly used BRDFs such as modified Phong and Ward BRDF. All such definitions and techniques provide necessary background for the literature review about rendering techniques including path tracing, photon mapping, and many-light rendering using virtual point lights in the next chapter.

2.1 Radiometry

2.1.1 Radiance

In computer graphics, physically based rendering is built upon radiometry, an area of study that deals with physical measurements of light [Dutre et al. 2006]. The goal is to compute the amount of light that travels and bounces in a given scene and is finally measured by a light measurement device. The physics term for this amount of light is called *radiance*, and is defined as follows.

In radiometry, flux (or radiant power, or power) is the power of light of a specific wavelength emitted from a source. It expresses light energy per unit time at a surface. Flux is denoted as Φ, and its unit is watt (W). Irradiance is the incident flux per unit area of a surface:

$$E(\mathbf{x}) = \frac{\mathrm{d}\Phi_i(\mathbf{x})}{\mathrm{d}A(\mathbf{x})}. \tag{2.1}$$

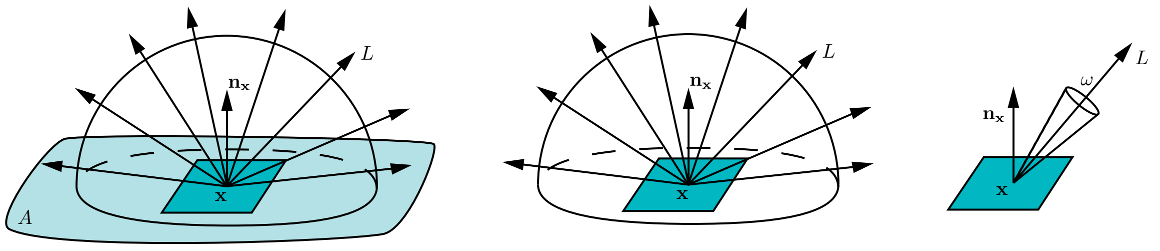


Figure 2.1: From left to right: flux, radiosity, and radiance.

The unit of irradiance is $W \cdot m^{-2}$. Similarly, radiosity or exitance radiance is the outgoing flux per unit area:

$$B(\mathbf{x}) = \frac{\mathrm{d}\Phi_o(\mathbf{x})}{\mathrm{d}A(\mathbf{x})}, \tag{2.2}$$

and its unit is also $W \cdot m^{-2}$. Radiance is the flux per solid angle per projected unit area.

$$L(\mathbf{x}, \omega) = \frac{\mathrm{d}^2\Phi(\mathbf{x})}{\mathrm{d}\omega \mathrm{d}A^{\perp}(\mathbf{x})} = \frac{\mathrm{d}^2\Phi(\mathbf{x})}{\mathrm{d}\omega \mathrm{d}A(\mathbf{x}) \cos\theta}. \tag{2.3}$$

The unit of radiance is $W \cdot sr^{-1} \cdot m^{-2}$ (watt per steradian per squared meter). Given the above definitions, we can easily relate radiance and irradiance by

$$\mathrm{d}E(\mathbf{x}) = L(\mathbf{x}, \omega) \cos\theta \mathrm{d}\omega. \tag{2.4}$$

Figure 2.1 further illustrates how outgoing flux and radiosity relate to radiance in terms of mathematical integration. Basically, outgoing flux is the integration of the outgoing radiance over the hemisphere and over the whole surface area; radiosity is the integration of the outgoing radiance over the hemisphere.

Human perceives brightness that can be expressed by radiance. In other words, radiance captures the look and feel of a scene that forms a picture to human eyes. In physically based rendering, our goal is to compute radiance at each surface that travels towards the light measurement device. In the next section, we would see that this process could be mathematically formulated as the *rendering equation.* In addition, to be concise, we generally refer to light measurement device as sensor, which can be an eye, a pinhole camera, or a camera with a lens and an aperture.

2.1.2 Invariance of radiance in homogeneous media

In the absence of participating media, the radiance along the ray that connects point $\mathbf{x}$ and point $\mathbf{y}$ is invariant. The energy conservation property can be derived as follows. The flux (watt) from $\mathbf{x}$ to $\mathbf{y}$ is

$$\Phi(\mathbf{x} \to \mathbf{y}) = \int_{A_\mathbf{y}} \int_{\Omega_\mathbf{x}} L(\mathbf{x} \to \mathbf{y})(\cos\theta_\mathbf{y} \mathrm{d}A_\mathbf{y}) \mathrm{d}\omega_\mathbf{x}, \tag{2.5}$$

where $\mathrm{d}\omega_\mathbf{x}$ is the solid angle subtended by the area at point $\mathbf{x}$ as seen from point $\mathbf{y}$ and can be computed as

$$\mathrm{d}\omega_\mathbf{x} = \mathrm{d}A_\mathbf{x} \cos\theta_\mathbf{x} / \|\mathbf{x} - \mathbf{y}\|_2^2. \tag{2.6}$$

Therefore, we have

$$\Phi_\mathbf{x} = \int_{A_\mathbf{y}} \int_{A_\mathbf{x}} L(\mathbf{x} \to \mathbf{y})(\cos\theta_\mathbf{y} \mathrm{d}A_\mathbf{y})(\mathrm{d}A_\mathbf{x} \cos\theta_\mathbf{x} / \|\mathbf{x} - \mathbf{y}\|_2^2). \tag{2.7}$$

Similarly, we can derive the flux from $\mathbf{y}$ to $\mathbf{x}$ as

$$\begin{aligned} \Phi(\mathbf{y} \to \mathbf{x}) &= \int_{A_\mathbf{x}} \int_{\Omega_\mathbf{y}} L(\mathbf{y} \to \mathbf{x})(\cos\theta_\mathbf{x} \mathrm{d}A_\mathbf{x}) \mathrm{d}\omega_\mathbf{y} \\ &= \int_{A_\mathbf{x}} \int_{A_\mathbf{y}} L(\mathbf{y} \to \mathbf{x})(\cos\theta_\mathbf{x} \mathrm{d}A_\mathbf{x})(\mathrm{d}A_\mathbf{y} \cos\theta_\mathbf{y} / \|\mathbf{y} - \mathbf{x}\|_2^2). \end{aligned} \tag{2.8}$$

Applying the energy conservation law, we have $\Phi(\mathbf{x} \to \mathbf{y}) = \Phi(\mathbf{y} \to \mathbf{x})$, and it is easily to deduce that the radiance along the ray is invariant, and we get $L(\mathbf{x} \to \mathbf{y}) = L(\mathbf{y} \to \mathbf{x})$.

2.1.3 Solid angle

Solid angle is defined by the projected area of a surface onto the unit hemisphere.

$$\mathrm{d}\omega = \frac{\mathrm{d}A(\mathbf{y}) \cos\theta_\mathbf{y}}{\|\mathbf{y} - \mathbf{x}\|^2}, \tag{2.9}$$

where $\mathbf{y} = h(\mathbf{x}, \omega)$. Function $h(\mathbf{x}, \omega)$ finds the nearest surface point that is visible to $\mathbf{x}$ from direction ω. Figure 2.2 illustrates the solid angle subtended by an arbitrary small surface located at $\mathbf{y}$ as seen from a small surface at $\mathbf{x}$.

In spherical coordinates, the differential solid angle is expressed as the differential area on the unit hemisphere:

$$\mathrm{d}\omega = (\sin\theta \mathrm{d}\phi)\mathrm{d}\theta = \sin\theta \mathrm{d}\theta \mathrm{d}\phi, \tag{2.10}$$

where θ and ϕ are the elevation and azimuth angle of the direction ω, and $\theta \in [0, \pi/2]$ and $\phi \in [0, 2\pi]$.

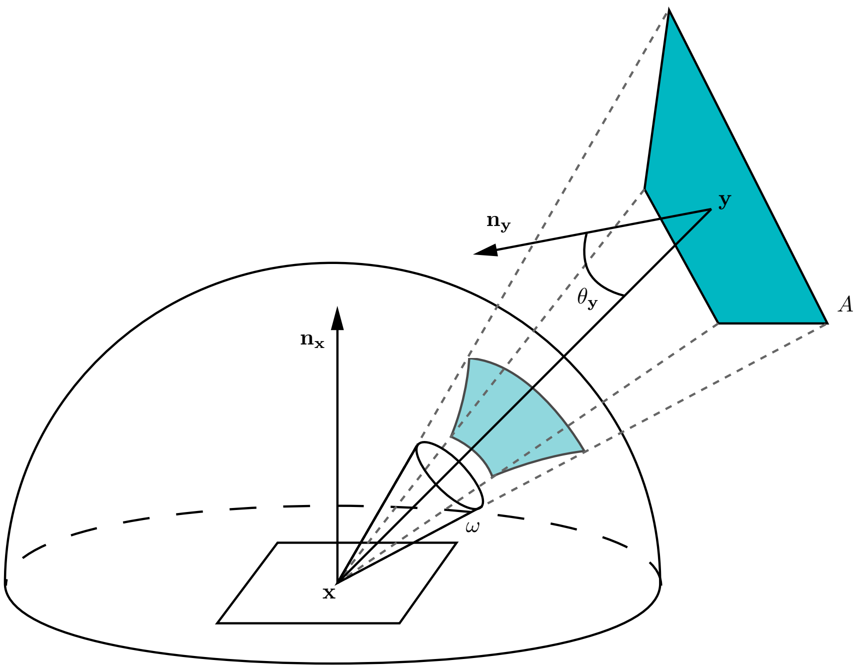


Figure 2.2: Solid angle.

2.1.4 The rendering equation

Given the above definitions, we are now ready to explain the rendering equation and its related terms. The rendering equation in the solid angle form is as follows:

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega} L_i(\mathbf{x}, \omega_i) f_s(\omega_i, \mathbf{x}, \omega_o) \cos\theta_i \mathrm{d}\omega_i, \tag{2.11}$$

where

- $f_s(\mathbf{x}, \omega_i, \omega_o)$: the bidirectional scattering distribution function (BSDF).
- $L(\mathbf{x}, \omega_o)$: the outgoing radiance at location $\mathbf{x}$ to direction ω_o.
- $L_i(\mathbf{x}, \omega_i)$: the incident radiance from direction ω_i to location $\mathbf{x}$.
- $L_e(\mathbf{x}, \omega_o)$: the emitted radiance at location $\mathbf{x}$ to direction (ω_o).

If we define the tracing function $h(\mathbf{x}, \omega)$ that returns the hit point $\mathbf{y}$ by tracing ray $(\mathbf{x}, \omega)$ into the scene, we can relate the incident radiance and outgoing radiance by

$$L_i(\mathbf{x}, \omega_i) = L(h(\mathbf{x}, \omega_i), -\omega_i). \tag{2.12}$$

This suggests that the above rendering equation is in fact defined in a recursive manner. The stopping condition of the recursion is when the ray hits a light source so it carries the emitted radiance from the light source.

The BSDF function determines how a ray interacts at a surface. Generally, a ray can either reflects at the surface or transmits into the surface depending on the physical properties of the surface. For example, when a ray hits a mirror, plastic, or diffuse surface, it reflects, while if it hits a glass, or prism, it bends and goes into the surface. As it is difficult to have a closed-form formula that supports all types of surfaces, in practice, for each material type of a surface, we bind it with a specific bidirectional scattering function. Functions that governs reflectivity of a ray is generally referred to as bidirectional reflectance distribution function (BRDF). Several BRDF models have been proposed in the literature, and we are going to explore a few popular models such as Phong BRDF and Ward BRDF in Section 2.3.

Given a camera model, for example, pinhole, the value of a pixel on the image plane can be calculated by integrating radiance of rays originating from the camera over the support of the pixel:

$$I(\mathbf{u}) = \int L_i(\mathbf{e}, \omega) W(\mathbf{e}, \omega) \mathrm{d}\omega, \tag{2.13}$$

where $\mathbf{u}$ is the pixel location, $\mathbf{e}$ the camera location, $\omega = \frac{\mathbf{u}-\mathbf{e}}{\|\mathbf{u}-\mathbf{e}\|}$, and W is the camera response function. Note that all points are defined in world space.

From the above equation, we see that it is necessary to estimate the radiance L in order to determine the value of the pixel. Therefore, radiance is the key value to manipulate in physically based rendering. In homogeneous media, e.g., air, glass, we assume radiance is invariant along a ray. To generate an image, our goal is to compute the radiance at each surface that reflects to each pixel on the image plane of the camera. The radiance can be found by performing integration as defined in the rendering equation. There are two popular techniques to solve the rendering equation, the Monte Carlo method, and finite element method. In the scope of this article, we are going to focus on Monte Carlo techniques to solve the rendering equation.

2.1.5 The area integral

Beside the solid angle form, the rendering equation can also be describe in the area form. In order to do so, imagine light that travels from a point $\mathbf{x}$ to $\mathbf{x}'$, reflects at $\mathbf{x}'$ and travels to $\mathbf{x}''$ as in Figure 2.3. The area integral can be written as

$$L(\mathbf{x}' \to \mathbf{x}'') = \int_{\mathbf{x}} L(\mathbf{x} \to \mathbf{x}') f_s(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') G(\mathbf{x}, \mathbf{x}') V(\mathbf{x}, \mathbf{x}') \mathrm{d}A(\mathbf{x}), \tag{2.14}$$

where $V(\mathbf{x}, \mathbf{x}')$ is a binary function which returns 1 only if $\mathbf{x}'$ is visible from $\mathbf{x}$. $G(\mathbf{x}, \mathbf{x}')$ is the geometry term that depends on the locations and orientations of both surfaces at $\mathbf{x}$ and $\mathbf{x}'$:

$$G(\mathbf{x}, \mathbf{x}') = \frac{\cos\theta_{\mathbf{x}} \cos\theta_{\mathbf{x}'}}{\|\mathbf{x} - \mathbf{x}'\|^2} = -\frac{\mathbf{n}_{\mathbf{x}}^\top (\mathbf{x} - \mathbf{x}') \mathbf{n}_{\mathbf{x}'}^\top (\mathbf{x} - \mathbf{x}')}{\|\mathbf{x} - \mathbf{x}'\|^4}. \tag{2.15}$$

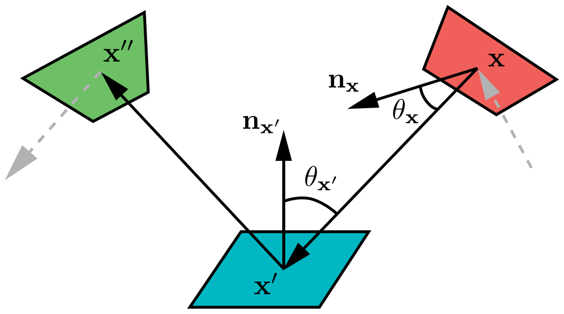


Figure 2.3: Three-point light transport.

It is also easy to convert the rendering equation between the area form and the solid angle form. Let $\omega_i = \mathbf{x} - \mathbf{x}'$, $\omega_o = \mathbf{x}'' - \mathbf{x}'$, $\mathrm{d}\omega_i = \mathrm{d}A(\mathbf{x})\cos\theta_{\mathbf{x}}/\|\mathbf{x} - \mathbf{x}'\|^2$, and assume that $\mathbf{x} = h(\mathbf{x}', \omega_i)$ so that $V(\mathbf{x}, \mathbf{x}') = 1$, we can transform the area integral into the solid angle form as

$$L(\mathbf{x}', \omega_o) = \int_\Omega L_i(\mathbf{x}', \omega_i) f_s(\omega_i, \mathbf{x}', \omega_o) \cos\theta_{\mathbf{x}'} \mathrm{d}\omega_i. \tag{2.16}$$

Note that in the above formula, $\theta_{\mathbf{x}'}$ is exactly the same as angle θ_i in the solid angle form of the rendering equation in Section 2.1.4.

2.1.6 The path integral

Veach [1998] described a non-recursive form of the rendering equation, which he named it the *path integral*, as follows.

$$L = \int_\Omega f(\bar{x}) \mathrm{d}\mu(\bar{x}), \tag{2.17}$$

where $\bar{x} = \mathbf{x}_0 \ldots \mathbf{x}_k$ is a path of length k, Ω the space of all paths of all lengths, $\mathrm{d}\mu(\bar{x}) = \mathrm{d}A(\mathbf{x}_0) \cdots \mathrm{d}A(\mathbf{x}_k)$, f the measurement contribution function:

$$\begin{aligned} f(\bar{x}) =& L_e(\mathbf{x}_0 \to \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \\ & \cdot \left(\prod_{i=1}^{k-1} f_s(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \right) \\ & \cdot W(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k). \end{aligned} \tag{2.18}$$

In this formulation, a path of length k expresses the light transport from a point on a light source that bounces $k-1$ times in the scene before reaching the light sensor. The value of a pixel is the integration of all paths of length from 0 to ∞ over the pixel support. The path integral is the fundamental theory for bidirectional path tracing [Veach 1998].

2.2 Monte Carlo integration

In general, it is not possible to derive an analytical formula for the integral in the rendering equation. Therefore, numerical methods are often used to evaluate the integral. Quadrature methods such as trapezoidal, mid-point, or Runge-Kutta rules work well for integrals of which domains are low dimensional. However, these methods perform badly when approximating the integral in the rendering equation due to its high dimensional nature. Fortunately, numerical methods that bases on randomization, which is often known as Monte Carlo methods, work very well for solving the rendering equation. We explore Monte Carlo integration in this section.

2.2.1 Monte Carlo estimator

Suppose that we would like to compute the following integral:

$$I = \int_{\Omega} f(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{2.19}$$

where Ω is the domain of $\mathbf{x}$. The Monte Carlo estimator of the integral is a function that produces an approximation value of I:

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}, \tag{2.20}$$

where $\mathbf{x}_i$ is a sample drawn from domain Ω with probability $p(\mathbf{x}_i)$, N is the total number of samples. Note that $p(\mathbf{x})$ is a probability density function, so $\int_{\Omega} p(\mathbf{x})\mathrm{d}\mathbf{x} = 1$. All the samples are drawn independently. It is easy to verify that the expected value of the estimator $\langle I \rangle$ is

$$\begin{aligned} \mathrm{E}\left[\langle I \rangle\right] &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}\left[\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}\right] = \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right] \\ &= \mathrm{E}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right] = \int_{\Omega} p(\mathbf{x})\frac{f(\mathbf{x})}{p(\mathbf{x})} = I. \end{aligned} \tag{2.21}$$

The *error* of the estimation is

$$\epsilon = \langle I \rangle - I. \tag{2.22}$$

The *bias* of a Monte Carlo estimator is defined as the expected value of the error, which can also be interpreted as the difference between the expected value of the estimator and the groundtruth value as follows:

$$\beta = \mathrm{E}\left[\epsilon\right] = \mathrm{E}\left[\langle I \rangle - I\right] = \mathrm{E}\left[\langle I \rangle\right] - I. \tag{2.23}$$

When the expected value of Monte Carlo estimator $\langle I \rangle$ is equal to I, the estimator $\langle I \rangle$ is called *unbiased.* The variance of the estimator is

$$\begin{aligned} \mathrm{V}\left[\langle I \rangle\right] &= \mathrm{V}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}\right] = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{V}\left[\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}\right] \\ &= \frac{1}{N}\mathrm{V}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right]. \end{aligned} \tag{2.24}$$

The mean squared error (MSE) of the estimator is

$$\mathrm{MSE} = \mathrm{E}\left[\epsilon^2\right] = \mathrm{E}\left[\langle I \rangle^2\right] + I^2 - 2 \cdot I \cdot \mathrm{E}\left[\langle I \rangle\right]. \tag{2.25}$$

Notice that since $\mathrm{V}\left[\langle I \rangle\right] = \mathrm{E}\left[\langle I \rangle^2\right] - \mathrm{E}\left[\langle I \rangle\right]^2$, it is easy to derive that

$$\mathrm{MSE} = \mathrm{V}\left[\langle I \rangle\right] + \beta^2. \tag{2.26}$$

When the estimator $\langle I \rangle$ is unbiased, its MSE is equal to the variance. The MSE convergence rate is $O(1/N)$, and the error rate is $O(1/\sqrt{N})$, which means in order to halve the error, a quadruple of current number of samples are needed.

Importance sampling

If we can choose $p(\mathbf{x}) = \frac{f(\mathbf{x})}{b}$ where b is a constant that ensures $p(\mathbf{x})$ is a probability density function, the variance of the estimator becomes

$$\mathrm{V}\left[\langle I \rangle\right] = \frac{1}{N}b, \tag{2.27}$$

which is a constant when N is fixed and converges to zero when $N \to \infty$. Theoretically, the normalization constant b can be computed by

$$b = \int_{\Omega} f(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{2.28}$$

The constant b can be estimated using Monte Carlo estimation using a few samples of $\mathbf{x}$ in the domain. However, we do not have the distribution $f(\mathbf{x})$ for the entire domain since it is what we would like to estimate. The best we can do is to choose $p(\mathbf{x})$ such that it resembles $f(\mathbf{x})$ as much as possible. The process of sampling $\mathbf{x}$ using such a distribution $p(\mathbf{x})$ is called *importance sampling.*

Rejection sampling

Generally, when $p(\mathbf{x})$ is standard or simple enough, $\mathbf{x}$ can be chosen by transforming from a uniform sample using some closed-form expressions. However, if such a transformation is too expensive to implement or does not exist, *rejection sampling* can be used. Suppose that the distribution function $p(\mathbf{x})$ can be bounded by a function $q(\mathbf{x})$ for all $\mathbf{x}$:

$$p(\mathbf{x}) < kq(\mathbf{x}), \tag{2.29}$$

where k is a scale value. Suppose that the distribution $q(\mathbf{x})$ is easier to sample than $p(\mathbf{x})$. A sample $\mathbf{x}$ can be generated as follows. First, sample $\mathbf{x}$ from distribution $q(\mathbf{x})$, and compute the probability $p(\mathbf{x})$ and $kq(\mathbf{x})$. Second, generate a uniform random number $u \in [0, kq(\mathbf{x})]$ and test if $p(\mathbf{x}) < u$. If true, reject $\mathbf{x}$ and repeat sampling $\mathbf{x}$. It can be seen that a sample $\mathbf{x}$ is accepted with probability $p(\mathbf{x})/kq(\mathbf{x})$ and thus overall the distribution of $\mathbf{x}$ follows $p(\mathbf{x})$. Rejection sampling can be slow as several samples might be drawn from $q(\mathbf{x})$ before one is accepted.

Quasi Monte Carlo methods

Deterministic sample results in aliasing. Random sample turns aliasing into noise. However, generating random numbers with a pseudorandom generator can result in numbers that are not well distributed. Quasi Monte Carlo methods work the same way as Monte Carlo estimators, but use special sequences, for example, Halton sequence, to generate the samples instead of relying on random sampling. The result is that variance can be reduced and the estimated result is less noisy. In this thesis, we would focus on Monte Carlo estimation.

2.2.2 Solving the rendering equation with Monte Carlo estimators

By applying Monte Carlo estimation, we can derive estimators for the rendering equation. For simplicity, we drop the emission term L_e as it does not affect the way the integral is approximated. In the solid angle form, suppose that at a point $\mathbf{x}$, we can sample direction ω according to a probability distribution $p(\omega)$. The estimator of the radiance from $\mathbf{x}$ to an outgoing direction ω_o is

$$\langle L(\mathbf{x}, \omega_o) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(\mathbf{x}, \omega_i) f_s(\omega_i, \mathbf{x}, \omega_o) \cos\theta_i}{p(\omega_i)}. \tag{2.30}$$

In this estimation, incoming direction ω_i is sampled using $p(\omega)$. To estimate $L_i(\omega, \omega_i)$, we continue to expand the above equation recursively. This suggests a simple ray tracing algorithm as follows. A ray can be traced from the camera origin through a pixel towards the scene and a surface point can be determined. A direction can be sampled at the surface point, and a new ray can be generated at that direction. The next hit point can thus be determined. The process can be repeated until a light source is hit. At each hit point, the outgoing radiance can be estimated using the above equation. This is a simple form of *path tracing* [Kajiya 1986].

Different pixels can have different rays and paths that hit light sources, and this variance appears as noise in the result. When more samples are used per pixel, noise is averaged out and disappear gradually and the estimated value converges to the exact integral value. The convergence speed depends on how good the sample is. We want to pick a sample such that it has a high contribution to the integral. Ideally, we would achieve constant variance (or no noise) if we could choose incident direction ω_i such that $p(\omega_i)$ is proportional to the product of incident radiance and BRDF in the integral. Unfortunately, this is not practical because such distribution is not available. In fact, such distribution is what we want to estimate. However, we still can choose ω_i such that its distribution is proportional to one of the terms in the product $L_i(\mathbf{x}, \omega_i) f_s(\omega_i, \mathbf{x}, \omega_o) \cos\theta_i$. We discuss various importance sampling techniques to sample light source, material, and geometric surfaces later in this chapter.

Similarly, the estimator for the area form can be written as

$$\langle L(\mathbf{x}' \to \mathbf{x}'') \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{L(\mathbf{x}_i \to \mathbf{x}') f_s(\mathbf{x}_i \to \mathbf{x}' \to \mathbf{x}'') G(\mathbf{x}_i, \mathbf{x}') V(\mathbf{x}_i, \mathbf{x}')}{p(\mathbf{x}_i)}. \tag{2.31}$$

In this form, instead of sampling incident direction ω_i, at each surface point $\mathbf{x}'$ that we want to evaluate the outgoing radiance, we sample point $\mathbf{x}$ on a surface in the scene and evaluate the contribution from $\mathbf{x}$ to $\mathbf{x}'$. Figure 2.3 demonstrates the light transport that flows from $\mathbf{x}$ to $\mathbf{x}'$, reflects at $\mathbf{x}'$ towards $\mathbf{x}''$. Notice that for each sample $\mathbf{x}$, we need to check the visibility between $\mathbf{x}'$ and $\mathbf{x}$. Since it is difficult to sample $\mathbf{x}$ so that $V(\mathbf{x}, \mathbf{x}')$ is always 1, this estimator can result in high variance, which means high noise in the rendered image. Therefore, in practice, the area form is seldom used to estimate radiance. The most common use of this form is to compute direct illumination by sampling light source surfaces or the environment map.

Lastly, the estimator for the path integral can be written as

$$\langle L \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{x}_i)}{p(\bar{x}_i)}. \tag{2.32}$$

In contrast to the solid angle and the area form, the path integral does not directly suggest how to sample a path. Several techniques to sample a path can be used. For example, a path of length k can be generated in $k + 2$ ways. We can trace all k segments of the path from the camera, or trace a few segments from the camera, a few from a light source, and connect the two endpoints together. Veach [1998] proposed multiple importance sampling, and the balance and power heuristics to account for different techniques to sample a path. These techniques are discussed in more details in Chapter 3.

2.3 Materials

Bidirectional scattering distribution function (BSDF) plays a key role in determining the appearance of surfaces in physically based rendering. It specifies the distribution of outgoing radiance when a light ray hits a surface from an incoming direction. Strictly speaking, at a particular surface point, BSDF is a 6D function which models the outgoing radiance according to the incident and outgoing direction, wavelength, and time. Here we assume BSDF is independent of wavelength and does not change over time, and thus BSDF can be modelled as a 4D function $f_s(\omega_i, \omega_o)$. Each surface has a BSDF that models its appearance.

Mathematically, the BSDF is the ratio of the outgoing radiance and the irradiance:

$$f_s(\omega_i, \mathbf{x}, \omega_o) = \frac{\mathrm{d}L(\mathbf{x}, \omega_o)}{\mathrm{d}E(\mathbf{x})} = \frac{\mathrm{d}L(\mathbf{x}, \omega_o)}{L(\mathbf{x}, \omega_i)\cos\theta_i \mathrm{d}\omega_i}. \tag{2.33}$$

BSDF can be modelled using mathematical formulas or expressed as tabulated data from measurements of real world materials. When only reflectance is considered, BSDF is also referred to as bidirectional reflectance distribution function (BRDF). For example, to model a glossy surface, analytical BRDF can be used such as modified Phong, Ward, and Ashikhmin model, which are empirical models, or Cook-Torrance, and He-Torrance model, which are based on reflection properties of physical surfaces. Empirical models are mathematical expressions with a few control parameters, which is simpler to sample and evaluate than physical models. Control parameters can be found by fitting the model to measured data from practice. In contrast, physical model is based on the concept of microfacet, which treats a surface as a cluster of many tiny flat surfaces that reflect light as a perfect mirror. While physical models are more expensive to sample and evaluate, they tend to provide more realistic rendering results. Recently, tabulated data is also widely adopted, for example, the MERL BRDF database [MERL 2006].

In general, there are two types of surfaces: metals and dielectrics. Metal conduct electricity; dielectrics do not. This property of electricity conductivity greatly decides the appearance of a surface.

There are two important properties to make a BSDF *physically plausible*: Helmholtz reciprocity and energy conservation.

Property 2.3.1. Helmholtz reciprocity: *the distribution of outgoing radiance is the unchanged if the incident and the outgoing direction is interchanged:*

$$f_s(\omega_i, \omega_o) = f_s(\omega_o, \omega_i). \tag{2.34}$$

which means if the lighting direction is reversed, the BSDF value remains the same. Several rendering algorithms such as path tracing, photon mapping, or bidirectional path tracing assumes Helmholtz reciprocity so that light paths can be traced both from light sources and camera.

Property 2.3.2. Energy conservation: *the total outgoing flux must be less than or equal to the total incident flux. This ratio is also referred to as* reflectance*:*

$$\rho(\mathbf{x}) = \frac{\int_\Omega L(\mathbf{x}, \omega_o) \cos\theta_{\omega_o} \mathrm{d}\omega_o}{\int_\Omega L(\mathbf{x}, \omega_i) \cos\theta_{\omega_i} \mathrm{d}\omega_i} \le 1. \tag{2.35}$$

This property can be rewritten by further expanding the term $L(\mathbf{x}, \omega_o)$ to

$$\frac{\int_\Omega \left(\int_\Omega L(\mathbf{x}, \omega_i) f_s(\omega_i, \omega_o) \cos\theta_{\omega_i} \mathrm{d}\omega_i\right) \cos\theta_{\omega_o} \mathrm{d}\omega_o}{\int_\Omega L(\mathbf{x}, \omega_i) \cos\theta_{\omega_i} \mathrm{d}\omega_i} \le 1. \tag{2.36}$$

From the above constraint, a necessary condition can be

$$\forall \omega_i : \int_\Omega f_s(\omega_i, \omega_o) \cos\theta_{\omega_o} \mathrm{d}\omega_o \le 1. \tag{2.37}$$

which means the integral of BSDF over the hemisphere should be bounded by 1.

In the next sections, we are going to discuss a few common BSDF models, including Lambertian, modified Phong, Ward, mirror, and glass. An important task when dealing with such BSDFs is to sample a direction ω_i from the surface where this BSDF attaches to, given an existing outgoing direction ω_o. Given a BSDF model, we choose to sample ω_i by following the distribution of the BSDF values $f_s(\omega_i, \omega_o)$.

2.3.1 The Lambertian model

Let $\rho(\omega_i, \omega_o)$ be the reflectance function which denotes the ratio of the outgoing flux to the incident flux. For the Lambertian model, ρ is a constant, and we have

$$f_s = \frac{\rho}{\pi}, \tag{2.38}$$

or the BRDF of a Lambertian surface is a constant and therefore independent of ω_i and ω_o.

Without considering the incoming radiance distribution, the best we can do is to sample this BRDF proportionally to the cosine of the elevation angle:

$$p(\omega) = \frac{\cos\omega}{\pi}. \tag{2.39}$$

Cosine-weighted sampling

Cosine-weighted sampling is a common strategy to generate rays in Monte Carlo approximation of the rendering equation. Recall that the rendering equation in solid angle form can be written as follows. Note that for simplicity, we can safely ignore the emitted radiance L_e from the current surface point and let the rendering equation be:

$$L(\mathbf{x}, \omega_o) = \int_\Omega f(\omega_i, \mathbf{x}, \omega_o) L_i(\mathbf{x}, \omega_i) \cos\theta_i \mathrm{d}\omega_i. \tag{2.40}$$

Let

$$p(\omega) = \frac{\cos\theta}{\pi}. \tag{2.41}$$

Note that $1/\pi$ is the constant to ensure $p(\omega)$ to sum to one. To sample ω, we change to polar coordinate system and sample the angles (θ, ϕ) such that

$$p(\theta, \phi) = \frac{\sin\theta\cos\theta}{\pi}, \tag{2.42}$$

where $\sin\theta$ accounts for the change of variables from ω to (θ, ϕ). The angles (θ, ϕ) can be generated from a pair of uniform random numbers (δ_1, δ_2) in $[0, 1)$ by using the formula:

$$\begin{aligned} \theta &= \sin^{-1}(\sqrt{\delta_1}), \\ \phi &= 2\pi\delta_2. \end{aligned} \tag{2.43}$$

The derivation of how to generate (θ, ϕ) from (δ_1, δ_2) is listed in the appendix at the end of this chapter.

Given the above probability distribution $p(\omega)$, the Monte Carlo estimator for the rendering equation can be simplified as follows:

$$\begin{aligned} \langle L(\mathbf{x}, \omega_o) \rangle &= \frac{1}{N} \sum_{k=1}^{N} \frac{f(\omega_k, \mathbf{x}, \omega_o) L_i(\mathbf{x}, \omega_k) \cos\theta_k}{p(\omega_k)} \\ &= \frac{\pi}{N} \sum_{k=1}^{N} f(\omega_k, \mathbf{x}, \omega_o) L_i(\mathbf{x}, \omega_k), \end{aligned} \tag{2.44}$$

where we use ω_k and θ_k to denote the k-th sample of ω_i and θ_i.

Finally, if the BSDF is Lambertian, and $f_s(\omega_i, \mathbf{x}, \omega_o) = \rho_\mathbf{x}/\pi$, the estimator is further simplified to

$$\langle L(\mathbf{x}, \omega_o) \rangle = \frac{\rho_\mathbf{x}}{N} \sum_{k=1}^{N} L_i(\mathbf{x}, \omega_k). \tag{2.45}$$

2.3.2 Modified Phong model

While Lambertian model has been widely used in the early days of computer graphics, it can only model diffuse materials. However, some materials in the real world are not diffuse, e.g., plastic and metal. Such glossy materials focus illumination into a narrow lobe instead of scattering illumination uniformly over the hemisphere as a diffuse material does. In

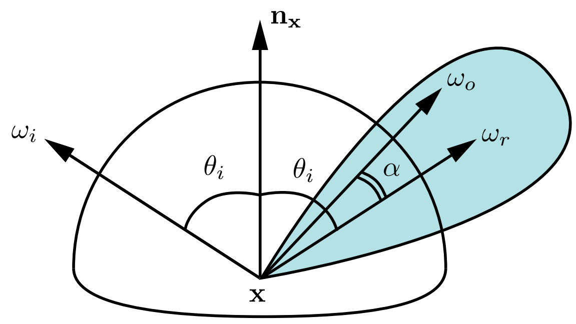


Figure 2.4: Sampling the Phong BRDF model.

addition, the appearance of glossy materials depend on the viewing angle. Therefore, more sophisticated models are required to render glossy surfaces.

The Phong model is an empirical BRDF that models isotropic glossy surfaces. It was first proposed by Phong [1975] and has been greatly used in among computer graphics community due to its simplicity and effectiveness. The modified Phong BRDF [Lafortune and Willems 1994] is similar to the Phong model, but with energy conservation property satisfied. It can be written as

$$f_s(\omega_i, \omega_o) = \frac{\rho_d}{\pi} + \rho_s \frac{n+2}{2\pi} \cos^n \alpha, \tag{2.46}$$

where α is the angle between the perfect mirror reflection direction ω_r and the outgoing direction ω_o, ρ_d the diffuse reflectance, ρ_s the specular reflectance when the incident ray is perpendicular to the surface, and n is the specular exponent. Figure 2.4 illustrates a specular lobe and the ray directions from the above formula.

We see that the Phong BRDF has into two components, diffuse and specular. The diffuse component follows the Lambertian model. The specular component is an exponential function that models the view-dependent specular highlight. The $(n+2)/(2\pi)$ is the normalization factor in order to ensure that the BRDF satisfies energy conservation.

The Phong BRDF satisfies Helmholtz reciprocity. To guarantee energy conservation, the total reflected energy needs to be smaller or at most equal to the incident energy, which can be expressed by

$$\rho_d + \rho_s \leq 1. \tag{2.47}$$

Sampling the modified Phong model

Sampling the Phong BRDF can be done by sampling the diffuse and specular component independently. As we have already known how to sample the Lambertian model with cosine-weighted sampling, here we will discuss how to sample the specular component in the Phong model. Given an incident direction, an efficient way to sample an outgoing direction in the specular lobe is to sample a direction about the perfect mirror direction. This can be done by sampling according to the distribution of the $\cos^n \alpha$ term:

$$p(\omega) = \frac{n+1}{2\pi} \cos^n \alpha, \tag{2.48}$$

where $(n+1)/(2\pi)$ is the normalization factor so that $\int_\Omega p(\omega) \mathrm{d}\omega = 1$. Given $p(\omega)$, the marginal probability $p(\theta)$ and $p(\phi \mid \theta)$, and the cumulative distribution $F(\theta)$ and $F(\phi \mid \theta)$ can be derived, from which we can sample a direction (θ, ϕ) from a pair of random numbers (δ_1, δ_2) by

$$\begin{aligned} \theta &= \arccos \delta_1^{\frac{1}{n+1}}, \\ \phi &= 2\pi\delta_2. \end{aligned} \tag{2.49}$$

Suppose that using the Phong BRDF, the radiance contributed by the diffuse component is L_d, and by the specular component is L_s. By sampling the diffuse component and the specular component separately, the estimator of the total radiance is

$$\langle L \rangle = \langle L_d \rangle + \langle L_s \rangle. \tag{2.50}$$

An outgoing direction for the diffuse component and an outgoing direction for the specular component needs to be sampled. This approach has no problems with direct illumination, but when computing global illumination, it can cause the number of rays to be generated to increase exponentially (two rays are generated at each ray-surface intersection). In fact, it is possible to estimate either L_d or L_s, without biasing the total radiance L so that we only need to generate one ray at a time. Given a constant τ, we can have

$$\langle L \rangle = \frac{\langle L_d \rangle}{\tau} \tag{2.51}$$

with probability τ, and

$$\langle L \rangle = \frac{\langle L_s \rangle}{1-\tau} \tag{2.52}$$

with probability $1-\tau$. The expected value of the estimator $\langle L \rangle$ is still unchanged and equal to the sum of expected value of $\langle L_d \rangle$ and $\langle L_s \rangle$. The constant τ can be chosen as

$$\tau = \frac{\rho_d}{\rho_d + \rho_s}, \tag{2.53}$$

which is the ratio of the diffuse relectance and the total maximum reflectance. This means more efforts are to be spent on estimating contribution from the component that reflects more energy. Note that the probability of a direction ω is now $p(\omega) = \tau p_d(\omega) + (1-\tau) p_s(\omega)$, which is the average of the probabilities of ω by sampling the diffuse and the specular component.

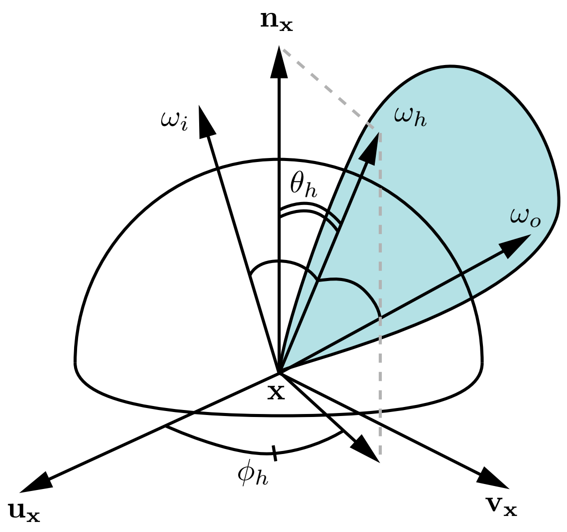


Figure 2.5: Sampling the Ward BRDF model based on the half vector ω_h.

2.3.3 Anisotropic Ward model

The Ward BRDF [Ward 1992] is an empirical model for anisotropic glossy surfaces, for example, brushed surfaces with streak highlight. The Phong BRDF can only model the appearance of plastic and metal of which the specular component is isotropic, where the highlight looks rounded and does not depend on rotations of the view about the surface normal.

The Ward BRDF can be written as follows.

$$f_s(\omega_i, \omega_o) = \frac{\rho_d}{\pi} + \frac{\rho_s}{4\pi\alpha_x\alpha_y\sqrt{\cos\theta_i\cos\theta_o}} e^{-\tan^2\theta_h\left(\frac{\cos^2\phi_h}{\alpha_x^2}+\frac{\sin^2\phi_h}{\alpha_y^2}\right)}, \tag{2.54}$$

where ω_h is the half vector between the incident and outgoing direction

$$\omega_h = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}. \tag{2.55}$$

Note that in the Phong model, since the specular highlight is isotropic, it is not necessary to strictly define the coordinate frame at the surface, i.e., the tangent can be chosen arbitrarily. For anisotropic gloss, a fixed coordinate frame must be used as it determines the orientation of the highlight. Figure 2.5 illustrates the local coordinate frame at the surface and the incident and outgoing directions.

An outgoing direction ω_o from the specular component of the Ward model can be sampled using the following probability [Walter 2005]:

$$p(\omega_o) = \frac{1}{4\pi\alpha_x\alpha_y\omega_h^\top\omega_i\cos^3\theta_h} e^{-\tan^2\theta_h\left(\frac{\cos^2\phi_h}{\alpha_x^2}+\frac{\sin^2\phi_h}{\alpha_y^2}\right)}. \tag{2.56}$$

To simplify the sampling process, we observe that ω_o can be derived from the half vector ω_h and the incident direction ω_i:

$$\omega_o = 2(\omega_i^\top\omega_h)\omega_h - \omega_i. \tag{2.57}$$

We can change the variables and sample the half vector ω_h instead using:

$$p(\omega_h) = p(\omega_o)(4\omega_h^\top\omega_i). \tag{2.58}$$

The full derivation can be found in the original work by Ward [1992] and in the technical report by Walter [2005]. In the local coordinate frame, direction ω_h can be generated from (θ_h, ϕ_h) by

$$\begin{aligned} \theta_h &= \arctan\sqrt{\frac{-\log\delta_1}{\cos^2\phi_h/\alpha_x^2+\sin^2\phi_h/\alpha_y^2}}, \\ \phi_h &= \arctan\left(\frac{\alpha_y}{\alpha_x}\tan(2\pi\delta_2)\right). \end{aligned} \tag{2.59}$$

2.3.4 Perfect mirror

Perfect mirror is a special case because given an incident direction, the outgoing direction can be determined exactly as $\omega_o = r(\omega_i)$ where

$$r(\omega) = 2(\omega^\top\mathbf{n})\mathbf{n} - \omega, \tag{2.60}$$

where $\mathbf{n}$ is the normal vector at the intersection. For a perfect mirror, the outgoing radiance is equal to the incident radiance.

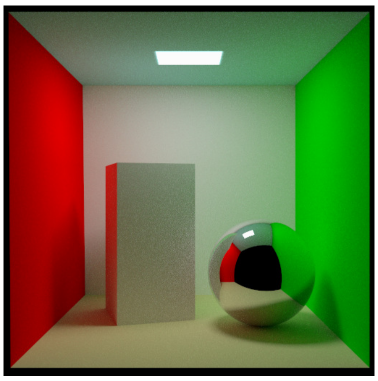
To avoid considering mirror as a special case, we can formulate a special BSDF for perfect mirror as follows:

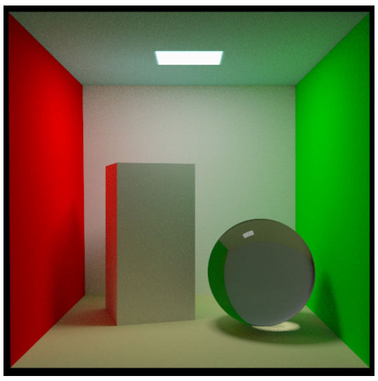
$$f_s(\omega_i,\omega_o) = \frac{\delta(\omega_i - r(\omega_o))}{\cos\theta_i}. \tag{2.61}$$

This function returns $1/\cos\theta_i$ when ω_o matches the mirror direction. The probability of the mirror direction is always 1.

2.3.5 Glass

Light does not always reflect when it hits a surface. When light hits the boundary between two media, it can refract and transmit from one medium to the other. Generally, the relation


(a) A mirror ball.


(b) A glass ball.

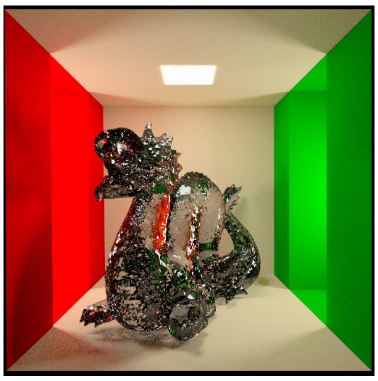

(c) A glass dragon.

Figure 2.6: The modified Cornell box.

between the incident and transmittance ray can be modelled by Snell's law:

$$n_i \sin\theta_i = n_t \sin\theta_t. \tag{2.62}$$

Given the incident direction ω_i, the transmittance direction ω_t can be computed as using the function $\omega_t = t(\omega_i)$ where

$$t(\omega) = -\frac{n_i}{n_t}\left(\omega - (\omega^\top \mathbf{n})\mathbf{n}\right) - \mathbf{n}\sqrt{1 - \frac{n_i^2}{n_t^2}(1 - (\omega^\top \mathbf{n})^2)}. \tag{2.63}$$

A special BSDF for glass material can be as follows.

$$f_s(\omega_i, \omega_o) = \frac{n_t^2}{n_i^2}\frac{\delta(\omega_i - t(\omega_o))}{\cos\theta_i}. \tag{2.64}$$

Note that due to refraction, the radiance of the transmittance ray does not equal to the radiance of the incident ray, but is scaled by n_t^2/n_i^2. This can be proved by using the energy conservation property at the refraction location. Intuitively, this can be explained by imagining a light ray that travels from air to water. In this case, the light rays refract and concentrate. Since the total energy is conserved, the concentrated rays account for a smaller volume, and so the transmitted rays have larger radiance value than incident rays. Similarly, when light goes from water to air, the transmittance rays have smaller radiance.

In fact, when a ray goes from a medium to another, it can both reflect and transmit at the intersection at the boundary. The ratio of light energy that reflects out of total incident light energy can be determined by the Fresnel equation. An estimation of the Fresnel equation is Schlick's approximation [Schlick 1994], which is much simpler to implement:

$$F = F_0 + (1 - F_0)(1 - \mathbf{n}^\top \omega_h)^5, \tag{2.65}$$

where F_0 is the reflected energy ratio when the incident light direction is parallel to the surface normal. F_0 can be computed as

$$F_0 = \left(\frac{n_i - n_t}{n_i + n_t}\right)^2. \tag{2.66}$$

Figure 2.6b presents the Cornell box scene with a glass sphere. The use of the Fresnel term here is important to render the reflection of the light source on the glass sphere.

2.4 Geometry

An object in a scene can be expressed as a set of surfaces. Surface is often represented in discrete form such as a set of triangles. Each triangle is defined by three points. Surface can also be described using implicit equation or parametric form. Parametric form of sphere or cylinder are often used to test ray tracing algorithms since it is relatively easy to compute intersection of a ray and a parametric surface.

Discrete surfaces such as triangles and rectangles and parametric surfaces such as disks and spheres are often used to construct scenes that are more complex. Global illumination algorithms such as ray tracing families are very flexible in handling both types of surfaces. However, when a rendering algorithm needs to be implemented on the GPU using the rasterization pipeline, parametric surfaces must be discretized into triangles before being transferred from the CPU memory to the GPU memory. This is because the rasterization pipeline has been specially designed to work with triangles and quadrilaterals. If the rendering algorithm is implemented using CUDA or OpenCL programming model and the rasterization pipeline can be entirely skipped, discretization is not necessary anymore and we can be more flexible in handling scene surfaces.

2.4.1 Octree

When tracing rays, an expensive operation is visibility check. Given point $\mathbf{x}$ and $\mathbf{y}$, checking whether $\mathbf{x}$ and $\mathbf{y}$ are visible to each other requires doing intersection test for the segment between $\mathbf{x}$ and $\mathbf{y}$ with all surfaces in the scene. However, checking against all surfaces in the scene is too expensive. Octree is a simple data structure that can speed up visibility check by limiting the intersection test to only relevant surfaces. Other data structures that can organize surfaces for fast visibility checking includes bounding volume hierarchy (BVH), and kd-tree. In this section, we discuss octree, as it is simple to implement.

The octree construction is a top-down process and can be described as follows. Each node in the tree has a bounding box attached to it. The root of octree is created with the bounding box of all surfaces in the scene. We keep splitting the bounding box of a node into eight and

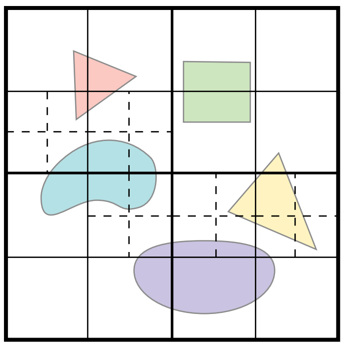


Figure 2.7: A 2D visualization of a quad-tree. Thickness of the border represents the level of a tree node. The thickest border represents the root.

create corresponding tree nodes until the bounding box of a node only contains a small set of surfaces.

Given the octree, checking intersection of a ray to a surface can be as follows. If the ray hits the bounding box stored by a tree node, continue to go down the tree and check if it hits any bounding boxes stored by the child nodes. If the ray does not hit a certain node, it is not necessary to check the intersection of the ray against its child nodes. Figure 2.7 demonstrates the idea of octree by its 2D representation, a quad-tree.

Tree data structures like octree, BVH, and kd-tree can be easily implemented on the CPU. However, they are not straightforward to implement on the GPU. When it is necessary to parallelize computation to achieve real-time frame rates, techniques for GPU ray traversal and visibility check can be used [Aila and Laine 2009].

2.4.2 Sampling basic shapes

In physically based rendering, beside organizing surfaces into octree or kd-tree for efficient ray-surface intersection check, another important task is to sample points on a surface uniformly. This operation is necessary for evaluating the area form of the rendering equation. Another common use of this operation is to sample points on an area light. In this section, we discuss strategies to efficiently sample points on surfaces modelled by basic shapes such as triangle and sphere.

Triangle

Given a triangle that is formed by three points $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$. In barycentric coordinate frame, a point $\mathbf{x}$ on the triangle can be represented by $\mathbf{x} = \alpha\mathbf{a} + \beta\mathbf{b} + (1 - \alpha - \beta)\mathbf{c}$, and $0 \leq \alpha \leq 1$

and $0 \le \beta \le 1 - \alpha$. To uniformly sample $\mathbf{x}$, α and β can be generated using

$$\begin{aligned} \alpha &= 1 - \sqrt{1 - \delta_1}, \\ \beta &= \delta_2 \sqrt{1 - \delta_1}. \end{aligned} \tag{2.67}$$

Sphere

A point on a sphere can be represented by (θ, ϕ) in spherical coordinate system and can be uniformly sampled by

$$\begin{aligned} \theta &= \arccos(1 - 2\delta_1), \\ \phi &= 2\pi\delta_2. \end{aligned} \tag{2.68}$$

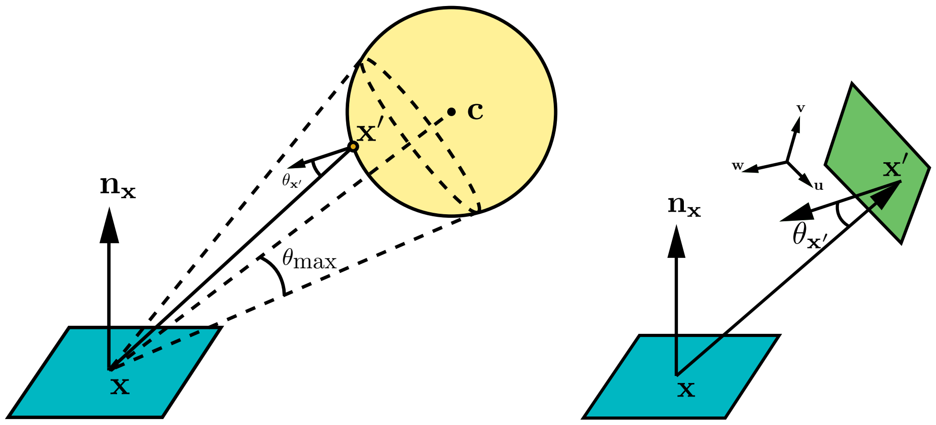
2.5 Light

Sampling a light source is very similar to sampling points on a surface that has a particular shape such as triangle, rectangles or sphere. However, uniform sampling on the surface of the light is not always the most efficient. The location of the receiver needs to be considered to reduce variance. In this section, we discuss the sampling of a sphere and a rectangle that can consider the location of the receiver. Wang's thesis [Wang 1994] and the article by Shirley et al. [1996] provides a thorough study about how to sample light source efficiently.

2.5.1 Spherical light

We discuss importance sampling for a light source that its shape is a sphere. To render direct illumination from a spherical light, a straightforward approach is to importance sample the sphere based on cosine-weighted sampling. However, this approach does not consider the location of the receiver surface, therefore, samples from the sphere can be occluded by the sphere itself, i.e., the samples lie on the hidden half of the sphere as viewed from the receiver. Shirley et al. [1996] proposed to uniformly sample rays in the cone subtended the sphere as viewed from the light source. This importance sampling approach works much better as it concentrates samples only in the visible light transport between the receiver and the light. Figure 2.8a illustrates this setting.

Assume that the sphere is has center $\mathbf{c}$ and radius r, and the location of the receiver is $\mathbf{x}$. We choose the local coordinate system to be at the receiver and the z-axis aligns to $\mathbf{c} - \mathbf{x}$. We can sample a point in the sphere by choosing a direction uniformly in the cone that bounds the sphere as viewed from the receiver. To perform cone sampling, we constraint the elevation angle θ to $0 \le \theta \le \theta_{\text{max}}$ where $\sin \theta_{\text{max}} = r / \|\mathbf{x} - \mathbf{c}\|$. The probability of a ray



(a) Spherical light. **(b)** Rectangular light.

Figure 2.8: Sampling spherical and rectangular light.

with direction ω sampled in the cone is

$$p(\omega) = \frac{1}{2\pi(1 - \cos\theta_{\max})}. \tag{2.69}$$

Calculating the marginal probability, the pair of elevation and azimuth angle (θ, ϕ) can be sampled using a pair of random numbers (δ_1, δ_2), both in $[0, 1)$, by

$$\begin{aligned} \theta &= \arccos(1 - \delta_1(1 - \cos\theta_{\max})), \\ \phi &= 2\pi\delta_2. \end{aligned} \tag{2.70}$$

2.5.2 Rectangular light

We seek to sample a point on a rectangular light source such that its probability is proportional to the amount of illumination it distributes to a receiver. Suppose that the coordinate system is at the lower left corner of the rectangle so that a point on it can be represented by $(u, v, 0)$. Let the receiver be located at $\mathbf{x} = (x_1, x_2, x_3)$ and the local surface orientation is $\mathbf{n}$. Figure 2.8b illustrates the configuration.

Suppose that we choose to sample (u, v) with the probability

$$p(\mathbf{x}') = p(u, v) = \frac{1}{C}\frac{\cos\theta'_{\mathbf{x}}}{\|\mathbf{x} - \mathbf{x}'\|^2}, \tag{2.71}$$

where C is the normalization constant that equals to

$$C = \int_0^{u_{max}} \int_0^{v_{max}} \frac{\cos\theta'}{\|\mathbf{x} - \mathbf{x}'\|^2}. \tag{2.72}$$

(u, v) can be generated by

$$\begin{aligned} F(v) &= \frac{1}{C} \arctan \frac{\hat{u}\hat{v}}{x_3\sqrt{\hat{u}^2 + \hat{v}^2 + x_3^2}} \Big|_{\hat{u}=-x_1}^{\hat{u}=u_{max}-x_1} \Big|_{\hat{v}=-x_2}^{\hat{v}=v-x_2} = \delta_1, \\ F(u|v) &= \frac{\frac{t}{\sqrt{t^2+(x_2-v)^2+x_3^2}} \Big|_{t=x_1}^{t=x_1-u}}{\frac{t}{\sqrt{t^2+(x_2-v)^2+x_3^2}} \Big|_{t=x_1}^{t=x_1-u_{max}}} = \delta_2, \end{aligned} \tag{2.73}$$

which needs to be solved numerically.

The thesis by Wang [1994] described in details several other probability distributions that (u, v) can be sampled from. While such distributions can greatly reduce the variance, they are too complex to implement efficiently in practice.

Chapter 3

Global illumination algorithms

This chapter describes fundamental global illumination algorithms and provides a literature survey for each algorithm. In the early days of computer graphics, global illumination was too costly to compute and often ignored. Nowadays, as CPU and GPU have been growing to be faster and cheaper, global illumination algorithms such as path tracing and photon mapping have been widely used in movie production and architectural visualization. Simple forms of global illumination, e.g., one-bounce interreflection, has also been added to real-time graphic engines. In this chapter, except the first section that is dedicated for direct illumination, we would be focusing on discussing global illumination algorithms.

3.1 Direct illumination

Direct illumination is the simplest type of rendering that illumination travels directly from light sources to surfaces, without bouncing to any other surfaces in between. In many cases, direct illumination accounts for a large amount of illumination of the scene. While rendering nowadays often includes indirect illumination, direct illumination still plays an important role in helping verify the correctness of scene geometry, light source and BSDF sampling, and thus is a great debugging tool for any rendering engines.

Based on the rendering equation, the direct illumination integral can be written as

$$L(\mathbf{x} \to \omega_o) = \int_{\mathcal{M}} L(\mathbf{y} \to \mathbf{x}) G(\mathbf{x}, \mathbf{y}) f_s(\mathbf{y} \to \mathbf{x} \to \omega_o) \mathrm{d}A(\mathbf{y}), \tag{3.1}$$

where $\mathcal{M}$ represents the set of all light surfaces in the scene. Recall that this integral is written in the area form, and can be estimated by Monte Carlo methods. A straightforward approach is to sample the light source surface and check the visibility between the sample and the receiver.

However, sometimes sampling the light source alone is not enough. Remember that the the integrand is the product $L(\mathbf{y} \to \mathbf{x}) G(\mathbf{x}, \mathbf{y}) f_s(\mathbf{y} \to \mathbf{x} \to \omega_o)$. Sampling the light source surface is a good way to achieve importance sampling on the distribution of incident illumination. However, this does not consider the location of the receiver and the value of the BSDF. To

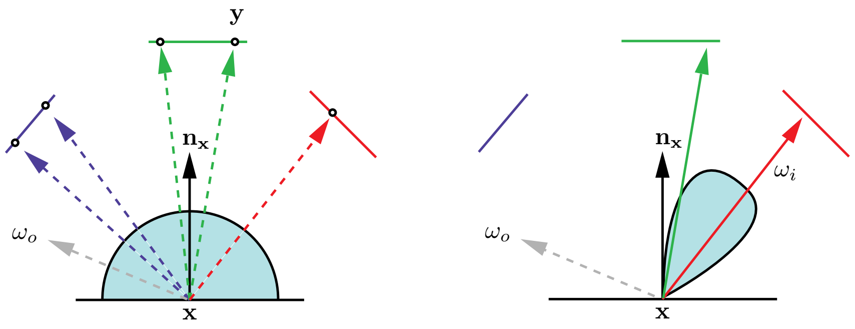


Figure 3.1: Sampling points on the light sources vs. sampling directions from the BSDF. Figure derived from [Gruenschloss et al. 2012] (see page 14).

sample independent distributions and combine them together, a good approach is multiple importance sampling (MIS) [Veach 1998].

3.1.1 Multiple importance sampling

Suppose that we need to evaluate the outgoing radiance contributed by a light source. There are two sampling strategies: sampling the light source and sample the BSDF. Each sampling strategy is good for a specific case, as illustrated in Figure 3.1. For example, when the surface is diffuse or the importance of incident rays are similar for all directions, sampling the light source is more efficient. It is not necessary to consider points that are not on any light sources, as they contribute no light to the receiver. In contrast, when the surface is glossy, sampling the BSDF is more efficient. While incident illumination can spread over a wide solid angle, only a small portion of it affects the outgoing radiance significantly.

Depending on which case our scene is, each sampling strategy can have its own advantages and disadvantages. Multiple importance sampling (MIS) can combine all strategies and weigh them in an unbiased manner. For example, in Figure 3.1, suppose that at the receiver $\mathbf{x}$ we evaluate the outgoing radiance using both strategies, which we denote as strategy A and B. Each strategy returns an estimation of the outgoing radiance. The weight for strategy A, using balance heuristics, is

$$w_A(\mathbf{y}) = \frac{p_A(\mathbf{y})}{p_A(\mathbf{y}) + p_B(\mathbf{y})}, \tag{3.2}$$

where p_A is the distribution used to generate $\mathbf{y}$, and $p_B(\mathbf{y})$ is the probability of $\mathbf{y}$ if it could have been generated using distribution p_B. Similarly, the weight for strategy B when it is used to generate a point $\mathbf{z}$ is

$$w_B(\mathbf{z}) = \frac{p_B(\mathbf{z})}{p_A(\mathbf{z}) + p_B(\mathbf{z})}. \tag{3.3}$$

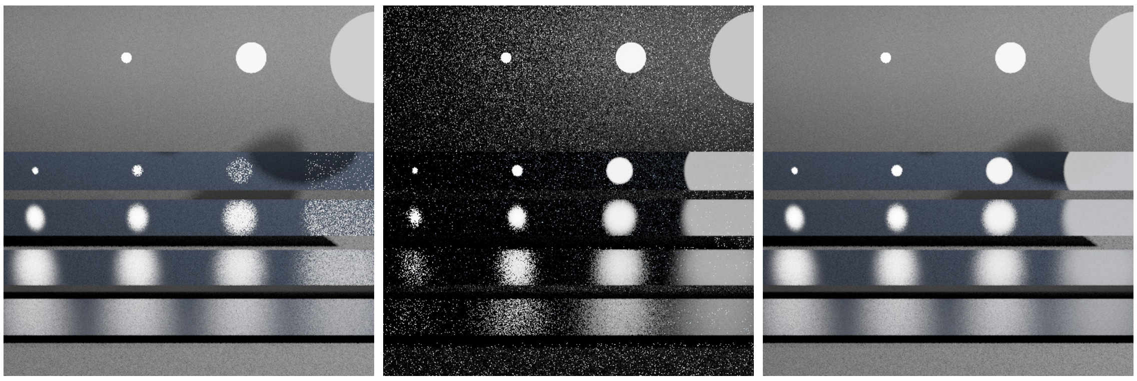

(a) Sampling light sources. **(b)** Sampling BRDFs. **(c)** MIS.

Figure 3.2: Multiple importance sampling. Images are rendered with 64 samples.

$p_A(\mathbf{z})$ expresses the chance to obtain $\mathbf{z}$ as if $\mathbf{z}$ were generated by sampling p_A distribution. The estimated radiance is

$$\langle L\rangle = \omega_A \langle L_A\rangle + \omega_B \langle L_B\rangle, \tag{3.4}$$

where $\langle L_A\rangle$ and $\langle L_B\rangle$ denotes the estimators that use strategy A and B, respectively. Intuitively, the balance heuristics assumes that the contribution to outgoing radiance should be large for samples that are generated with high probability. While this is not always the case, it works well in practice [Veach 1998].

The above example can be generalized to more than two sampling strategies as well as to the case in which the number of samples used in each strategy is not the same. As long as the weights over all strategies sums to one, the Monte Carlo estimator is unbiased. For a formal description of multiple importance sampling and proofs of the balance heuristics, see Chapter 9 of the thesis by Veach [1998]. Figure 3.2 illustrates a scene from Mitsuba [Jakob 2010] rendered with 64 samples. Multiple importance sampling is used to sample the specular highlights efficiently.

3.2 Unidirectional path tracing

Path tracing and light tracing can be referred to as unidirectional path tracing, as light path is always started and traced in a single direction, either from camera to light source or light source to camera.

3.2.1 Path tracing

Path tracing is a Monte Carlo unbiased rendering algorithm that has been widely used to generate reference images in physically based rendering. Path tracing can compute global illumination, and is relatively easy to implement.

In path tracing, a light path is sampled by establishing vertices incrementally from the camera towards light sources. The first vertex of the path is the camera location. By sampling a point on the image plane, a ray can be traced from the camera towards the scene. The second vertex of the path is the intersection of this ray with the scene. By sampling a new direction or a new surface point at the intersection, the third vertex can be determined, and this can be repeated. The path is complete when a vertex falls onto a light source.

The throughput of each path can be computed as keeping track of the geometry terms, visibility, and the BSDF values so far when each vertex of the path is sampled. Dividing the throughput by the probability of the path yields the radiance estimation. Each path is often referred to as a sample in Monte Carlo estimation. Averaging the estimated radiance over several samples is necessary to achieve a low-variance estimation of the rendering integral.

A path of length k estimates the $(k-1)$-bounce illumination. For example, a two-segment path yields an estimation of direct illumination; a three-segment path yields an estimation of one-bounce indirect illumination, and so on. Generally, path tracing does not limit the length of a path, but we can terminate a path after a few bounces. However, this is biased because illumination contributed by remaining bounces are not considered and all set to zero. To address this issue, Russian Roulette (RR) is an unbiased way to terminate a path. At each vertex, we choose to continue the path with probability α, and terminate with probability $1-\alpha$. The new estimator with Russian Roulette can be written as

$$\langle L_{RR} \rangle = \begin{cases} \langle L \rangle / \alpha & \text{if path is continued,} \\ 0 & \text{if path is terminated.} \end{cases} \tag{3.5}$$

It is easy to verify that the expected value of the estimator $\langle L_{RR} \rangle$ is the same as the estimator $\langle L \rangle$. The value of α can be chosen based on the average reflectance of surfaces in the scene or the local surface reflectance.

As it is rather costly to generate a path, vertices along the path can be reused to connect to light sources. This establishes new paths that each share the same vertices with the original path up to the vertex that is connected to light sources. The new paths can be correlated (strictly speaking, paths should be generated independently), but the efficiency gained is far more significant.

3.2.2 Light tracing

Light tracing works in the same way as path tracing except that light paths are generated from light sources towards the sensor. The second last vertex of a path is connected to the sensor.

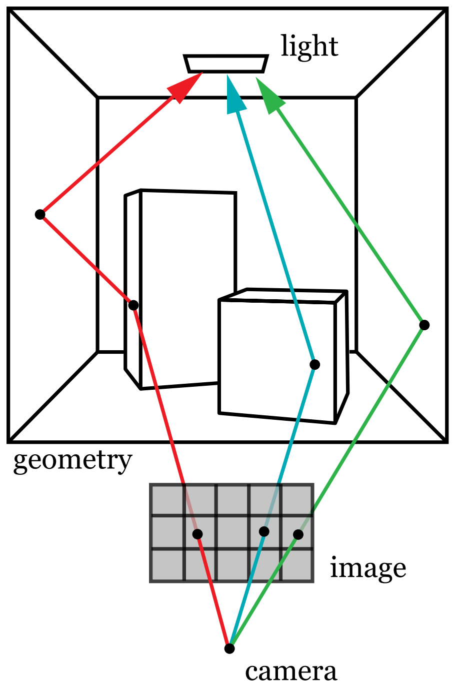


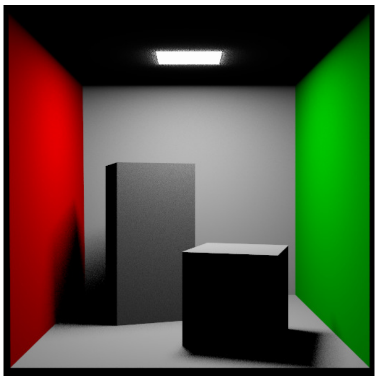
Figure 3.3: Path tracing.

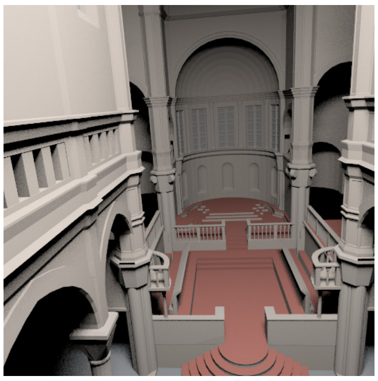
As light paths in path tracing are sampled from camera, path tracing is good at rendering specular highlights and mirror effects. In contrast, light tracing is better at finding caustics as caustics is the result of a light transport that contains a series of specular reflection or transmission events before the path ends by a diffuse reflection. Figure 3.5 demonstrates this advantage in light tracing. The caustics due to the concentration of light through a glass sphere can be rendered very quickly with light tracing, which is smooth with only 64 samples. Path tracing can render the caustics, but it is still noisy after more than 500 samples. However, the glass sphere cannot be rendered with light tracing because it is impossible for an eye ray that connects to the glass surface can match the transmission ray from a light subpath, so the BSDF is zero. In such cases, a combination of path tracing and light tracing is desirable to make the rendering of the glass sphere efficient.

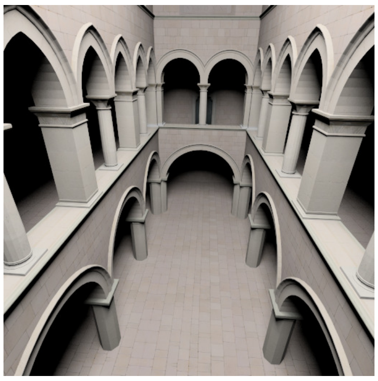
3.3 Bidirectional path tracing

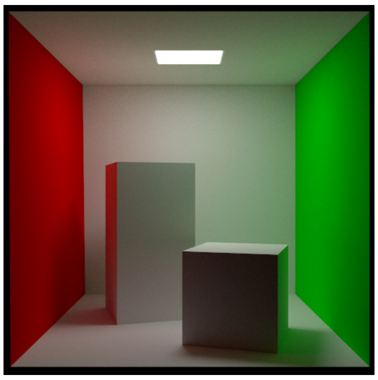
We follow the notation by Veach [1998] in describing bidirectional path tracing. In path space, the rendering equation can be written as a Lebesgue integral. It is named the *path integral formulation*:

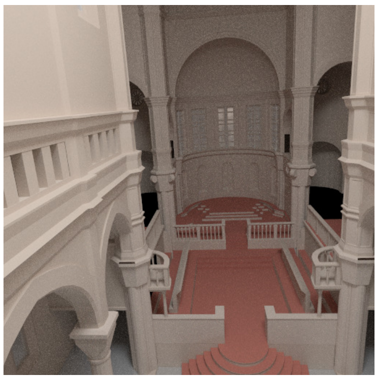
$$L = \int_{\Omega} f(\bar{x}) \mathrm{d}\mu(\bar{x}), \tag{3.6}$$


(a) The Cornell box.


(b) The Sibenik scene.


(c) The Sponza scene.


(d) The Cornell box.


(e) The Sibenik scene.

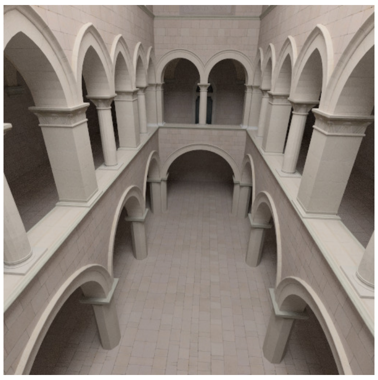
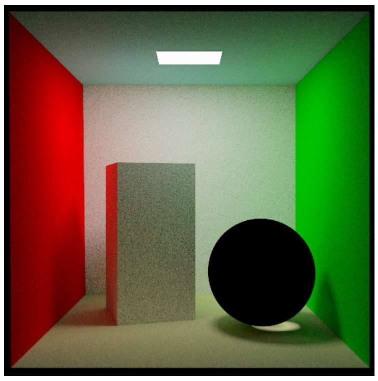

(f) The Sponza scene.

Figure 3.4: Direct illumination and global illumination. The second row is generated by path tracing. The Sibenik and Sponza scene are from [McGuire 2011].


(a)

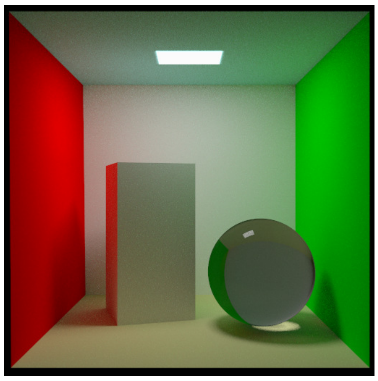

(b)

Figure 3.5: The modified Cornell box rendered by (a) light tracing and (b) path tracing. Note the smoother caustics with fewer samples in (a).

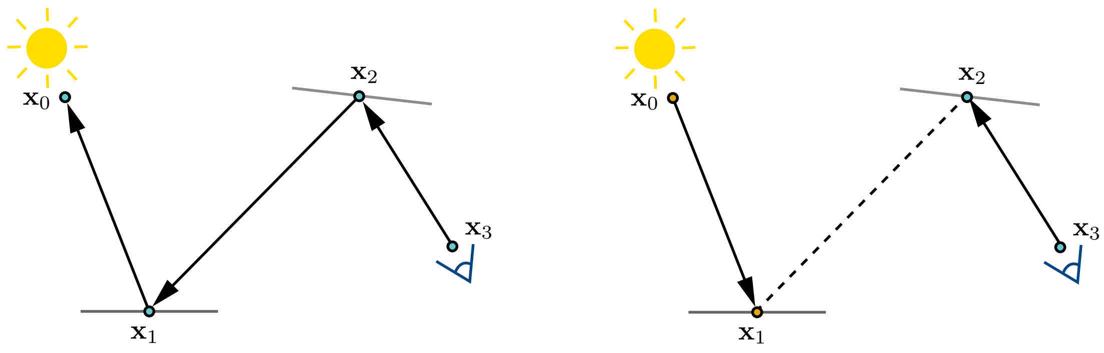


Figure 3.6: Different ways to generate a complete light path.

where $\bar{x} = \mathbf{x}_0 \ldots \mathbf{x}_k$ is a path of length k, Ω the space of all paths of all lengths, $\mathrm{d}\mu(\bar{x}) = \mathrm{d}A(\mathbf{x}_0) \cdots \mathrm{d}A(\mathbf{x}_k)$, f the measurement contribution function:

$$\begin{aligned} f(\bar{x}) =& L_e(\mathbf{x}_0 \to \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \\ & \cdot \left(\prod_{i=1}^{k-1} f_s(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \right) \\ & \cdot W(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k). \end{aligned} \tag{3.7}$$

Similar to path tracing, Monte Carlo sampling can be used to estimate the integral:

$$\langle L \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{x}_i)}{p(\bar{x}_i)}, \tag{3.8}$$

where the probability of each path can be defined as

$$p(\bar{x}) = \prod_{i=0}^{k} p(\mathbf{x}_i). \tag{3.9}$$

In bidirectional path tracing, light paths are generated by joining sub-paths started from light sources and eyes. By convention, a light transport path has the first vertex on a light source and the last vertex on the sensor. For example, in Figure 3.6, $\mathbf{x}_0\mathbf{x}_1$ is a sub-path that starts from a light source, and $\mathbf{x}_2\mathbf{x}_3$ is a sub-path that starts from the sensor. To create a full light path, the simplest way is to connect $\mathbf{x}_1$ with $\mathbf{x}_2$ to form the path $\mathbf{x}_0 \ldots \mathbf{x}_3$. The path $\mathbf{x}_0 \ldots \mathbf{x}_3$ can be generated by several ways as shown in Figure 3.6. During Monte Carlo estimation, this path can appear several times due to different sampling techniques. It is necessary to consider how this path could have been generated with other techniques and weigh its contribution in an unbiased manner. Veach [1998] assumed that contributions by paths generated with high probabilities are more important, and proposed two heuristics, the *balance* heuristics and *power* heuristics, to weigh the path contribution.

Given a light sub-path and an eye sub-path, several light transport paths of different length can be generated. For example, in Figure 3.6, it is also possible to connect $\mathbf{x}_0$ with $\mathbf{x}_2$, or $\mathbf{x}_1$ with $\mathbf{x}_3$, etc. To make good use of all sub-path vertices, we can consider all possible ways to create complete paths from the sub-paths. While the paths can be correlated, this is a good trade-off because paths are expensive to trace in the scene.

As bidirectional path tracing combines the strength of path tracing and light tracing into a single framework, it is able to render a wider range of effects including glossiness and caustics.

3.3.1 State of the arts in path tracing

Hachisuka et al. [2012] and Georgiev et al. [2012] proposed to combine bidirectional path tracing and photon mapping into a single framework using multiple importance sampling. The key idea is to formulate photon mapping as a path sampling technique. The MSE of this combined approach converges asymptotically at the rate $O(1/N)$ with carefully chosen parameters, which is as good as bidirectional path tracing. The new approach can efficiently handle specular-diffuse-specular (SDS) paths while retaining the benefits of path tracing.

Kaplanyan and Dachsbacher [2013a] showed that for paths that cannot be sampled by any techniques due to singularities, i.e., paths that arises from sampling a perfect mirror or a point source, regularization can be applied to make the paths become possible to sample. The central idea is to turn the singular sampling domain into a non-singular domain so that we can sample in it, and then gradually reduce the domain size after each iteration. The authors showed that progressive photon mapping can be regarded as a form of regularization. Similarly, virtual spherical light [Hašan et al. 2009] is also a regularization where the radii of the lights can be reduced to produce consistent estimations.

To evaluate the multidimensional rendering intergral, Hachisuka et al. [2008] proposed to use a kd-tree to distribute samples in multidimensional space. The largest variance leaf node is selected and a best-candidate sample is added to the leaf node. The leaf node is split if necessary, and the process repeats until a number of samples are reached. In the reconstruction step, anisotropic filtering is used to preserve edges. The weight of each sample is the volume of the kd-tree leaf node that the sample occupies.

To remove noise in Monte Carlo estimation more effectively, adaptive sampling and reconstruction can be used to distribute more samples into image regions of which errors are high. Li et al. [2012] proposed an approach to estimate error using Stein's unbiased risk estimator. This technique allows error estimation for anisotropic reconstruction kernel, which allows better preservation of high frequency details as compared to box filter or Gaussian kernel.

Another notable class of techniques that are relevant to path tracing and adaptive sampling is Metropolis light transport [Veach and Guibas 1997]. The goal of such techniques is to distribute paths more densely into brighter regions in the image. This is done by creating a new path that can be locally near an existing path using mutation techniques. Efficient mutation techniques such as lens mutation, caustics mutation, multi-chain mutation are proposed in [Veach 1998]. Recently, a mutation technique that is efficient for specular surfaces is proposed in [Jakob and Marschner 2012]. They showed that specular paths are confined in a low-dimensional manifold that can be explored more efficiently. In addition, Lehtinen et al. [2013] showed that the framework of Metropolis light transport can be changed to render image gradients. The final image can then be reconstructed by solving the Poisson equation.

3.4 Photon mapping

Photon mapping is a rendering technique that aims to estimate irradiance by the density of photons in a local area at the surface. This rendering method has two steps. The first step is photon tracing. Photons are emitted at light sources, and ray-traced by following light paths in the scene. At each intersection of the light rays and scene surfaces, a new photon is deposited. The power of each photon is approximately the same, therefore, the density of photons in a local neighborhood of a surface gives an approximation of the energy arriving at that surface. The second step is radiance estimation. Rays are generated from the camera. For each ray, at the first intersection with scene surfaces, the irradiance due to each photon is estimated by

$$E = \frac{\Phi}{\pi r^2}, \tag{3.10}$$

where the area of the surface is estimated by a disk with radius r centered at the receiver. Given the irradiance, the outgoing radiance at the receiver due to each photon can be estimated by

$$L(\mathbf{x} \to \omega) = E f_s(\omega_i \to \mathbf{x} \to \omega)(\mathbf{n}^\top \omega_i). \tag{3.11}$$

Photon mapping can be used to estimate indirect illumination, since indirect illumination is often smooth. Direct illumination can be computed independently by techniques discussed in Section 3.1.

A limitation in photon mapping is the memory needed to store photons in the first step. To estimate indirect illumination accurately, tens of millions of photons are required. Progressive photon mapping [Hachisuka et al. 2008] removes this limitation by adding new photons incrementally and computing the outgoing radiance contributed by this new set of photons through radius reduction of the density kernel. Stochastic photon mapping [Hachisuka and Jensen 2009] extended progressive photon mapping to estimate multidimensional rendering integral in order to support depth of field and motion blur effects. Knaus and Zwicker [2011] presented a probabilistic derivation of progressive photon mapping. It shows that it is not necessary to maintain statistics locally as the original progressive photon mapping work, i.e., the tracking of local photon density is unnecessary for radius reduction. In fact, radius can be reduced by a rate that is independent of the photon statistics. Recently, Kaplanyan and Dachsbacher [2013b] derived an optimal convergence rate for progressive photon mapping from the theory of regressions in statistics, and demonstrated how to perform radius reduction locally and adaptively.

The rendering cost in photon mapping depends on the ratio between the scene size and the smallest feature in the scene [Walter et al. 2012]. The approximation sphere cannot be larger than this smallest feature in order to resolve it. In large scenes, the sphere tends to be initialized with large radius, which can take a long time to reduce to the size that can resolve tiny features in the scene. Many-light rendering is a more efficient approach in this aspect. It can resolve details with fewer VPLs, but cannot be as robust as photon mapping in handling some types of effects such as caustics.

It has been well known that bidirectional path tracing samples specular-diffuse-specular (SDS) paths with low probability. Photon mapping and its progressive methods are more efficient in rendering such paths, but they have low order of convergence [Knaus and Zwicker 2011]. The maximum MSE convergence rate for bidirectional path tracing is $O(1/N)$, and for progressive photon mapping is $O(1/N^{2/3})$. Hachisuka et al. [2012] and Georgiev et al. [2012] proposed to combine bidirectional path tracing and photon mapping into a single framework using multiple importance sampling. The key idea is to formulate photon mapping as a path sampling technique. The MSE of this combined approach converges asymptotically at the rate $O(1/N)$ with carefully chosen parameters. For details, see [Georgiev et al. 2012].

3.5 Many-light rendering

One of the first work in many-light rendering is instant radiosity [Keller 1997], which proposes to approximate global illumination using a set of light particles. Such particles act as point lights that scatters illumination to the scene. But since the point lights do not exist in the physical world, they are called virtual point lights (VPLs). While instant radiosity assumes that surfaces are Lambertian, many-light rendering can be easily extended to render scenes with glossy surfaces by properly evaluating the BSDF at each the surface where each VPL is stored. Many-light rendering is a two-pass algorithm. In the first pass, light subpaths are traced from light sources and at each vertex of the subpath, a VPL is generated. In the second pass, eye subpaths are traced from the sensor. Similarly, at each vertex of the eye subpath, a virtual sensor point (VPS) can be stored. The last vertex of each eye subpath is connected to each last vertex of each VPL to create complete light paths. The formulation of VPL rendering is as follows.

Consider a complete path of ℓ segments which can be generated by connecting a light sub-path of s segments to an eye sub-path of t segments. Denote the vertices of the path as $\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_s\mathbf{z}_t \dots \mathbf{z}_1\mathbf{y}_0$ where $\mathbf{y}_i$ and $\mathbf{z}_j$ are vertices on light and eye sub-path, respectively. In the first pass of VPL rendering, when a light sub-path is traced, at each light vertex $\mathbf{y}_i$ ($i \in 0 \dots s$), a VPL is stored. Each VPL denotes a light sub-path $\mathbf{y}_0 \dots \mathbf{y}_i$. Similarly, at each eye vertex $\mathbf{z}_j$, a VPS is stored. Each VPS represents an eye sub-path $\mathbf{z}_j \dots \mathbf{z}_0$. The radiance of the path generated by connecting a VPL $\mathbf{y}$ to a VPS $\mathbf{z}$ is

$$L(\mathbf{y}_i \to \mathbf{z}_j \to \mathbf{z}_{j-1}) = \frac{T_i T_j}{p(\bar{y}_i)p(\bar{z}_j)} f_s(\mathbf{y}_{i-1} \to \mathbf{y}_i \to \mathbf{z}_j) G(\mathbf{y}_i, \mathbf{z}_j) f_s(\mathbf{y}_i \to \mathbf{z}_j \to \mathbf{z}_{j-1}), \tag{3.12}$$

where T_i and T_j are the throughput of the light and eye sub-path, respectively. Instant radiosity can now be viewed as a special case of VPL rendering, where eye paths are of length $t = 1$. In this case, an eye sub-path is fixed and has only two vertices $\mathbf{z}_1$ and $\mathbf{z}_0$ and the throughput of the eye sub-path is $T_j = 1$. The above equation can be simplified to

$$L(\mathbf{y}_i \to \mathbf{z}_1 \to \mathbf{z}_0) = \frac{T_i}{p(\bar{y}_i)} f_s(\mathbf{y}_{i-1} \to \mathbf{y}_i \to \mathbf{z}_1) G(\mathbf{y}_i, \mathbf{z}_1) f_s(\mathbf{y}_i \to \mathbf{z}_1 \to \mathbf{z}_0). \tag{3.13}$$

To keep our discussion easier to follow, we would assume the length of eye sub-path to be 1 from now on, and only revert to the general case if necessary.

3.5.1 Generating VPLs and VPSes

VPLs can be generated by sampling light sources and tracing light subpaths. For each vertex of the path, a VPL is generated. The VPL represents the light subpath of which the last vertex is at the VPL location. A VPL records throughput, probability of the subpath, together with the incident direction and the BSDF of the surface on which the last vertex stays.

The simplest approach to generate VPLs is to sample light sources and trace paths that start from light sources. The subpath can be terminated using Russian Roulette. This approach is almost similar to light tracing except that the last vertex of the subpath is not connected to the sensor until later in the gathering pass. One drawback of this approach is that the generation of the VPLs do not consider the sensor location. Some VPLs may be wasted if it is occluded and cannot reach the sensor.

Similarly, VPSes can be generated by tracing eye subpaths. In [Walter et al. 2012], short eye sub-paths with few segments is preferred in order to control the number of VPSes since the number of VPSes depends on the image resolution that we need to render.

3.5.2 Gathering illumination from VPLs

In the gathering pass, each VPL is connected to all VPSes, or pixels on the sensor in the single segment eye-path case. This gathering step is quite similar to light tracing or the

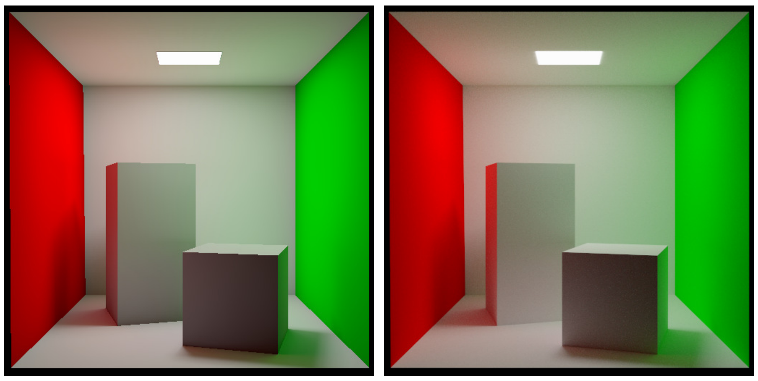

(a) ≈ 3000 VPLs. (b) Reference (path tracing).

Figure 3.7: The Cornell box rendered by many-light rendering.

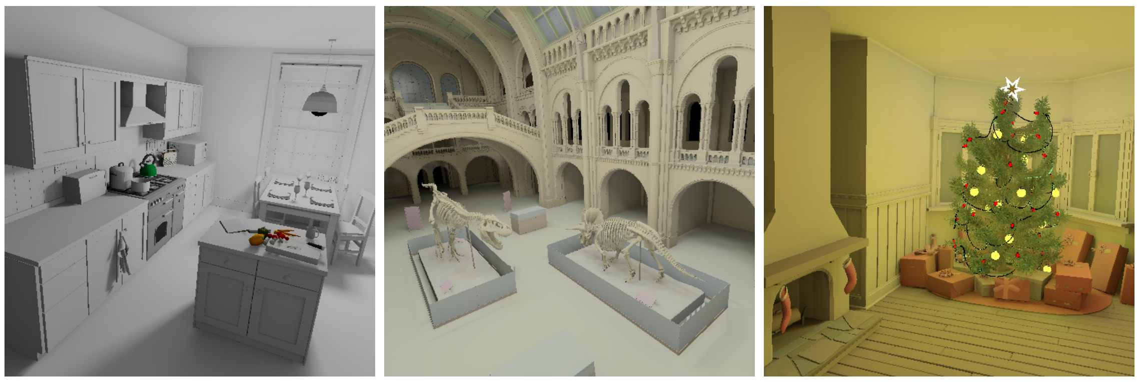

(a) The Kitchen scene. $\approx 42K$ VPLs. (b) The Natural History scene. $\approx 16K$ VPLs. (c) The Christmas scene. $\approx 300K$ VPLs.

Figure 3.8: Complex scenes rendered by many-light rendering. The Kitchen scene is from [Hardy 2012], the Natural History and the Christmas scene from [Birn 2014].

connections in bidirectional path tracing. In light tracing, the last vertex of the subpath is connected to the sensor to form a complete path, and the pixel that the path contributes to can be determined by intersecting the path with the image plane. Each pixel has independent set of light subpaths. In many-light rendering, light subpaths (which are VPLs) are shared among all pixels. This leads to high coherence in the estimated radiance, and the gathering step is very easy to parallelize and can be make use of the rasterization pipeline on the GPU.

The radiance from a VPL to every pixel can be implemented on the GPU using shaders. The visibility between a VPL and all pixels can be computed using shadow maps, which is fast. This is an advantage as compared to path tracing and bidirectional path tracing, of which the cost for ray tracing to perform visibility check dominates the total rendering time. Figure 3.7 demonstrates the Cornell box scene rendered using VPLs. Figure 3.8 illustrates the rendering of scenes that are more complex.

A shader implementation for many-light rendering

Assume that the set of VPLs is given. A standard two-pass algorithm for many-light rendering can be as follows. This can be implemented as fragment shaders.

- Render from the view of the VPL to record the shadow map.
- Render from the camera view. For each pixel, check if its gather point can be seen from the VPL by querying the shadow map. If it is visible, evaluate the VPL and accumulate the contribution to the pixel intensity.

We can also implement many-light rendering by off loading some computations to vertex shaders. Here are the steps.

- Render from the virtual point light point of view to record the depth map. In the vertex shader, the light intensity, and geometry term values are stored to each vertex. In the fragment shader, such values are interpolated and stored to a texture in additional to the standard shadow map.
- Render from the camera point of view. Perform shadow map lookup to evaluate visibility. For visible fragments, perform an additional texture lookup to retrieve the light intensity values and geometry term for each fragment. Perform a BRDF evaluation at the fragment to complete the radiance calculation.

This implementation shifts the evaluation of the BSDF at the VPL location and the geometry term calculation that is usually implemented in a fragment shader to the vertex shader of the first step, right before the shadow map is created. It stores the geometry term and BSDF values as attributes for each vertex, and relies on the graphics hardware to interpolate such values for each fragment in the rasterization. Therefore, shading details depends on the how the geometry is subdivided. While this is an approximation to the standard fragment shader implementation, it can be useful for real-time applications, especially when the VPLs are distant from the gathering surfaces and the BSDFs of the VPLs are diffuse.

3.5.3 Visibility query

Shadow mapping is one of the most common techniques to evaluate visibility from a point to all other points. Shadow mapping is easy to fit into the rasterization pipeline, and can be efficiently implemented on the GPU. It has been widely used for visibility test in many-light rendering.

Monte Carlo techniques such as bidirectional path tracing still relies on ray tracing to probe point-to-point visibility. It can be implemented on the CPU with acceleration structures such as bounding volume hierarchy (BVH), or kd-tree. It can also be paralellized on the GPU. The efficiency of ray tracers on NVIDIA GPUs is reported in [Aila and Laine 2009].

The emerging trend of ray tracing for real-time graphics have also created new interests in accelerating visibility query in ray tracing. Popov et al. [2013] proposed to cache visibility in a hash map. They assume visibility between two points can be approximated by visibility of the clusters that store the points. When visibility between two points is evaluated using ray tracing, the result is cached. Other visibility queries between the parent clusters can use the same cache value, and no further ray is traced.

3.5.4 Progressive many-light rendering

Progressive rendering is easy to implement in many-light rendering. In each frame, a subset of total VPLs are evaluated and their contributions are added into an accumulation buffer. For display, it is necessary to account for the missing energy of those VPLs that are not yet being evaluated by scaling the radiance values in the accumulation buffer properly. We can simply choose the scale factor to be the ratio between total VPLs and the number of VPLs evaluated so far. It is easy to see that this ratio converges to one when all VPLs are evaluated and therefore, no missing energy correction is required.

Dammertz et al. [2010] proposed to combine VPL rendering with caustics histograms and specular gathering into a single system to handle a wide range of illumination phenomena. In their method, VPLs are responsible for generating low frequency indirect illumination, and used as illumination source for specular gathering. In specular gathering, eye subpaths are traced until the paths hits a diffuse surface or terminated by Russian Roulette. Such eye subpaths are then connected to VPLs to build complete light paths to estimate specular illumination. They can also be used with photons in caustics histograms to complete caustics paths. Since the system is built upon VPLs, it inherits the progressive nature of VPL rendering.

3.5.5 Bias in many-light rendering

In principle, many-light rendering is unbiased as long as the VPLs are generated and evaluated by following the Monte Carlo framework strictly. In practice, this is not always the case. For example, the density of VPLs can be sparse at some locations, and when the VPLs appear to be too close to some gathering surfaces, bright spots appear in the final image, which is as much annoying to human perception as noise. To eliminate the bright spots, a workaround is to clamp the total illumination contributed by a VPL to a threshold. While this trick can clean up the final image, it causes bias. However, biased VPL rendering and photon mapping are two of the most commonly used biased algorithms in practice [Walter et al. 2012]. We investigate the bias problem in more details in Chapter 5.

3.5.6 Clustering of VPLs

In instant radiosity, the rendering cost is linear to the number of VPLs. Lightcuts [Walter et al. 2005] clusters the VPLs by building and traversing a binary tree and only evaluates the representative light in each cluster to reduce the rendering cost to sublinear. Similar complexity is also achieved by exploiting matrix clustering [Hašan et al. 2007]. We further investigate the clustering problem in Chapter 4.

3.5.7 Glossy surfaces

Gathering from VPLs in order to render glossy surfaces is generally not efficient. This is because the VPLs can be too sparse that they do not sample the specular lobe in the glossy BRDFs well. We render a scene provided by Mitsuba renderer that is modelled after the multiple importance sampling test scene in [Veach 1998] and observe the specular highlights in the scene. Figure 3.9 illustrates the scene rendered by gathering a number of VPLs.

In this scene, the power of the four light sources that are close to the metal plates are equal to each other. Therefore, VPLs are generated uniformly from these four light sources according to power sampling (using power as a probability distribution to sample light sources). This can be observed in the specular highlights on the plates as the VPLs try to fill such highlight regions. The size of the highlights reflects the size of light sources. As VPLs are generated uniformly, highlights of big light sources are more difficult to generate as more VPLs are needed to fill in such large highlight regions. Figure 3.9a demonstrates the appearance of the scene rendered by $50K$ VPLs. Even after such large amount of VPLs, the highlights still cannot be rendered correctly. This case study shows that high gloss reflection of large objects are difficult to render using VPLs. To solve this issue, an eye pass is needed to sample glossy BRDFs efficiently.

To test this possibility, a eye path of length two is traced for each pixel, and VPLs are generated and stored only if the eye path hits a light source. Given the image size of 512×512, about $20,000$ VPLs are generated. This way of VPL sampling ensures that the glossy BRDF can be efficiently sampled and for each visible surface point, there is a higher chance that some VPLs fall into the specular lobe of the BRDF. Figure 3.9 demonstrates the results of this simple experiment.

However, a disadvantage of this VPL sampling approach is that the generation of VPLs now depends on image resolution and how camera rays are generated. Davidovič et al. [2010] proposed to share eye-path VPLs in a local neighborhood. Therefore, in their approach, a local VPL only contributes to a number of surfaces that appears to be close to each other in the image space. Bidirectional lightcuts Walter et al. [2012] is an approach that is built on top of VPL rendering and it additionally connects particles starting from eye to the lights. Therefore, radiance estimated by VPLs and eye particles can be combined using

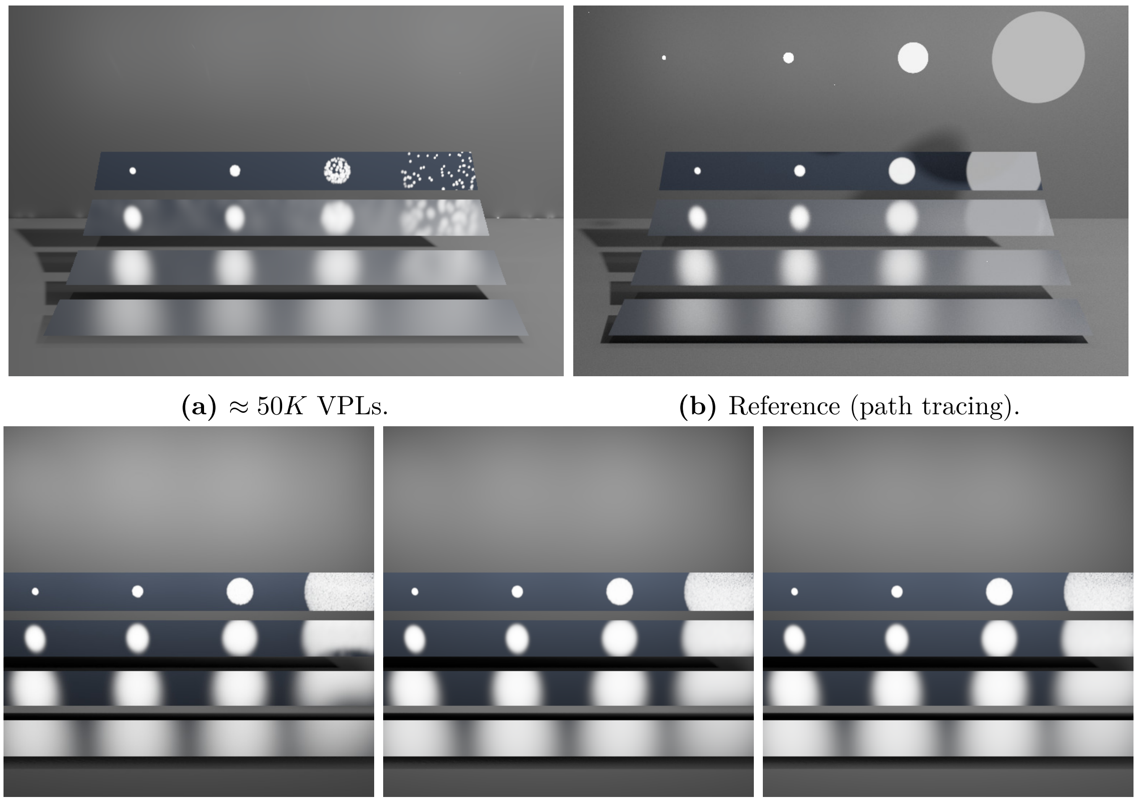

(a) $\approx 50K$ VPLs.
(b) Reference (path tracing).
(c) $\approx 10K$ VPLs.
(d) $\approx 16K$ VPLs.
(e) $\approx 22K$ VPLs.

Figure 3.9: The gathering process with VPLs generated by tracing (a) light paths and (c)-(e) eye paths of length two.

multiple importance sampling. Multidimensional lightcuts [Walter et al. 2006] are applied to efficiently evaluate VPLs and eye particles.

3.6 Interactive and real-time global illumination

In addition to offline rendering, many-light rendering has been widely adapted to rendering at interactive and real-time frame rates. One of the earliest and common technique is reflective shadow map [Dachsbacher and Stamminger 2005]. This approach supports one-bounce indirect illumination by sampling VPLs in shadow maps of light sources. Visibility tests between VPLs and shading points can be ignored, or implemented using standard shadow maps. Imperfect shadow maps [Ritschel et al. 2008] is an extension on this step with visibility approximation. It builds low-resolution shadow maps from the approximation point cloud of the original scene geometry. The transformation and projection of a point cloud is much faster than the rasterization of triangles and polygons. Shadow maps of a point cloud can contain holes, which is imperfect as compared to shadow maps of the original geometry.

The holes can be filled by interpolation and thus possible to render thousands of imperfect shadow maps per frame. This approach works well for smooth indirect illumination.

Implicit visibility [Dachsbacher et al. 2007] is a reformulation of the rendering integral so that visibility between surface points can be ignored. This leads to extra flux to be transferred among surfaces, which is compensated by a negative amount of energy, called antiradiance. The authors showed that finite element discretization similar to radiosity could be used to solve the new rendering integral. GPU implementation is possible for this technique, but handling glossy materials can be difficult due to discretization.

Micro-buffer rendering [Ritschel et al. 2009] further explores parallelism in rendering one-bounce indirect illumination. At each gather point, a micro frame buffer that has very low resolution, i.e., 8×8 to 24×24 is generated. Each micro buffer can be regarded as a mapping from the unit hemisphere above the gather point. Each micro pixel therefore corresponds to an incident direction and a solid angle. The micro pixel value stores the incident radiance from the direction it represents. To fill the micro buffer, a point hierarchy of the scene geometry is traversed to determine the nearest visible surface point and its illumination to each micro pixel. Given the micro buffer, the reflected radiance at the gather point is simply the sum over all micro pixels. Due to low-resolution micro buffer, this approach can only renders diffuse and rough glossy indirect illumination.

Techniques based on splatting indirection illumination from VPLs to image pixels are also proposed. Dachsbacher and Stamminger [2006] splats a quadrilateral. The quadrilateral is computed by bounding the volume of surfaces that a VPL can contribute to. A tighter bound by an eclipse that is discretized into a spherical triangle mesh can be used to limit the number of pixels a VPL needs to splat to. Essentially, the bound covers the region where the illumination from the VPL is significant, for example, larger than a threshold. Nichols and Wyman [2009] proposed to splat illumination to a multi-resolution buffer. A splat is subdivided adaptively into subsplats based on screen space discontinuities. Each subsplat is rendered into a layer in the multi-resolution buffer. Which layer a subsplat should be is determined based on the size of the subsplat.

Tokuyoshi and Ogaki [2012] demonstrated how to implement bidirectional path tracing on the GPU with rasterization. Their method supports at most two-bounce indirect illumination (length-4 paths). Eye subpaths are traced incrementally using global ray bundles. At each visible surface point, a local direction is sampled and used for all rays in next event estimation. This is the key that allows eye subpath tracing to be implemented with rasterization using only a perspective projection and a parallel projection. Light subpaths are generated using reflective shadow maps. However, their method only perform samples global directions uniformly and does not handle BRDF importance sampling.

Global illumination can also be approximated in image space. The most popular form is ambient occlusion, which is the average visibility count from a surface point to all other surfaces. Screen-space ambient occlusion (SSAO) methods approximate ambient occlusion using visibility data in a depth map. Ritschel et al. [2009] extends SSAO to include directional occlusion and one bounce of indirect illumination. It makes uses of direct illumination stored in image space to approximate the indirect illumination.

Kaplanyan and Dachsbacher [2010] proposed to store flux from VPLs sampled in reflective shadow maps in 3D grids and propagate flux among the grids. The flux distribution in each grid cell is represented as spherical harmonics coefficients. After propagation, the final flux in each cell can be used to evaluate the reflected radiance at each surface gather point. This approach can render smooth indirect illumination and participating media. Loos et al. [2011] also studied light propagation, but among canonical shapes such as cubes and cylinders. Precomputed radiance transfer is performed on the canonical shapes. To render a complex scene, its geometry is mapped to canonical shapes so that the precomputed transfer data can be reused. This method is designed to scale from high-end to mobile platforms. However, since it relies on precomputation, it is not suitable for dynamic scenes.

Many-light on the GPU also suffers from bias due to clamping. Novák et al. [2011] proposed screen-space bias compensation. Since it only uses image space data, it cannot consider objects out of the camera view. Photon mapping has also been ported to the GPU. Mara et al. [2013] studied a set of techniques to implement photon mapping efficiently on the GPU. They concluded that the best technique so far is to assign photons into screen-space tiles and perform final gathering for pixels in a tile using photons that belong to that tile. This does not require a data structure such as kd-tree to query k-nearest neighbors in the gathering pass, which is more difficult to implement on the GPU. A hash grid that is a hash table built on a 3D grid can also be used to group photons. A gather point that is in a 3D grid cell can use all photons in the cell for final gathering.

3.7 Conclusions

This chapter provides a literature survey of recent advances in global illumination algorithms. The techniques can be approximately categorized into path tracing and bidirectional path tracing, Metropolis light transport, photon mapping, and many-light rendering. Each of these techniques has pros and cons, which should be carefully considered before being integrated into an application. For example, path tracing is a robust technique that has been used widely in movie production. It is simple to develop and does not have a lot of parameters to tweak. Photon mapping is especially efficient to render caustics while virtual point light is good for a noiseless interactive preview. Metropolis light transport is very robust in adaptive sampling but it requires more effort to implement.

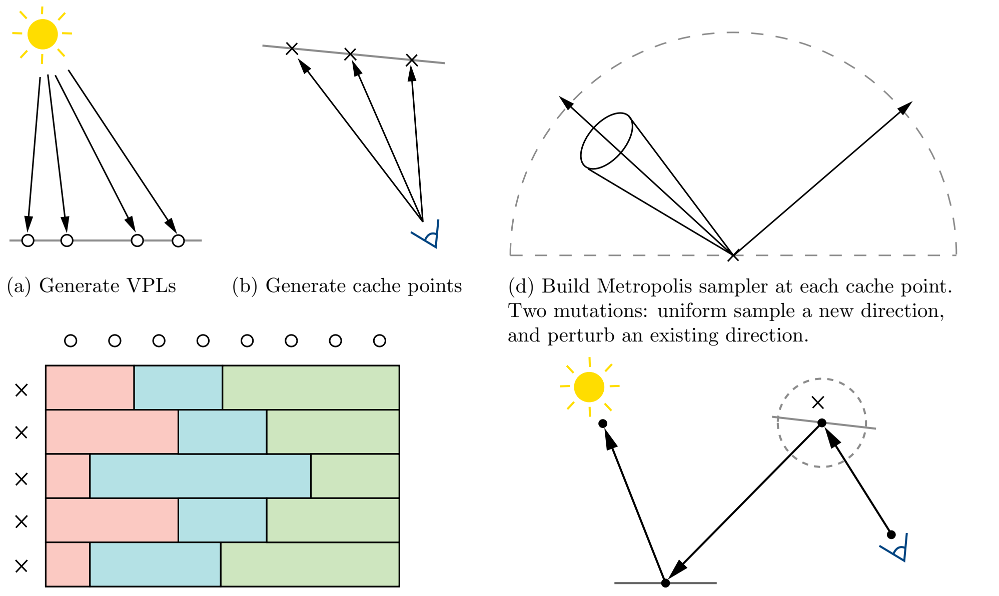
In the next chapter, we are going to explore how the effectiveness of path tracing can be improved by further utilizing virtual point lights for importance sampling.

Chapter 4

Guided path tracing using virtual point lights

From our previous discussion, many-light rendering with VPLs can be easily implemented on the graphics hardware and run at interactive frame rates. This method is also scalable if the VPLs are clustered before evaluation. Such benefits bring more capability to applications such as architectural and lighting design, movie production, where previewing is an important task. Unfortunately, VPL has a few drawbacks. For example, its gathering step is known to cause singularity, which appears as bright splotches in the result. In addition, it is challenging to use VPL to render glossy surfaces effectively. Several works have been proposed to overcome these drawbacks. For example, one might first render low-frequency effects using VPL gathering, and then apply path tracing to estimate high-frequency effects [Kollig and Keller 2006; Davidovič et al. 2010]; or evaluate VPL in a way that artifacts can be avoided [Hašan et al. 2009]. Such approaches aim to modify VPL so that it can both avoid artifacts and incorporate glossy appearance rendering into a unified framework.

In this chapter, we explore an application of VPLs in Monte Carlo path tracing. In contrast to VPL, path tracing can easily render a wide range of materials including glossy appearance. However, the noise in path tracing may take a while to disappear, and this process depends on the efficiency of the sampling techniques used to construct light paths. We propose to utilize clustered virtual point lights to improve the effectiveness of path tracing. To achieve this goal, we propose a Metropolis algorithm to sample directions from the unit hemisphere that utilizes the incoming radiance estimated by the VPLs. We also make our approach scalable by incorporating VPL clustering into it. Our experiments show that our Metropolis sampler can improve the effectiveness of importance sampling in path tracing, especially for diffuse surfaces. When it is combined with a BRDF sampler using multiple importance sampling, path tracing converges significantly faster.



(a) Generate VPLs (b) Generate cache points (d) Build Metropolis sampler at each cache point. Two mutations: uniform sample a new direction, and perturb an existing direction.

(c) Cluster VPLs (e) Path tracing. Importance sample a direction using the Metropolis sampler of a nearby cache point.

Figure 4.1: An overview of our approach. We sample directions based on the distribution of incoming radiance estimated by virtual point lights. The main steps of our approach is as follows. (a) A set of VPLs is first generated. (b) Surface points visible to camera are generated and grouped into clusters based on their locations and orientations. The representatives of the clusters are used as cache points which store illumination from the VPLs and guide directional sampling. (c) The light transport from the VPLs to the cache points are computed. To support scalability, for each cache point, the VPLs are clustered adaptively by following LightSlice [Ou and Pellacini 2011]. (d) We can now sample directions based on incoming radiance estimated by the VPL clusters. At each cache point, we store a sample buffer and fill it with directions generated by the Metropolis algorithm. (e) In Monte Carlo path tracing, to sample at an arbitrary surface point, we query the nearest cache point and fetch a direction from its sample buffer.

4.1 Related works

4.1.1 Many-light rendering

Instant radiosity [Keller 1997] is the seminal work that proposes to approximate indirect illumination using a set of point lights. Despite its great efficiency, instant radiosity can cause bright splotches and dark corners in the result, and is not able to render glossy surfaces and caustics effectively. A common approach to overcome artifacts is to clamp the reflectivity between a VPL and a gather point to a threshold level. Kollig and Keller [2006] proposed to estimate the missing energy due to clamping using path tracing. Dammertz et al. [2010] proposed a framework that unifies instant radiosity and caustics rendering into a single system.

There are also several variants of virtual point light. Hašan et al. [2007] proposed to evaluate VPL as spherical light to avoid artifacts and render low-frequency glossy surfaces. Novák et al. [2012b] proposed virtual ray light to support participating media rendering. Later, they also proposed virtual beam light [Novák et al. 2012a] which is the extension of virtual spherical light for participating media. Engelhardt et al. [2012] introduced approximate bias compensation for rendering heterogeneous participating media. An up-to-date survey of VPL techniques is recently published by Dachsbacher et al. [2014]. We refer our readers to this state of the art report for a more thorough discussion of the topic.

To make instant radiosity scalable, VPLs can be clustered so that only cluster representatives are evaluated at each gather point. Estimating global illumination with the clustered VPLs is also known as many-light rendering. Matrix row-column sampling [Hašan et al. 2007] is a technique that clusters columns of a sub-sampled light transport matrix formed by evaluating the VPLs for a subset of gather points. The clusters are then used as VPL clusters to evaluate outgoing radiance at all other gather points. Ou and Pellacini [2011] proposed to refine column clusters for each gather point, hence preserving local illumination more effectively. Davidovič et al. [2010] extends matrix row-column sampling to support glossy surface appearance. They proposed to generate VPLs from the gather points and use them to estimate the energy lost due to clamping. This is a variant of bias compensation using path tracing [Kollig and Keller 2006].

Another class of methods to cluster VPLs is lightcuts [Walter et al. 2005]. The VPLs are arranged into a binary tree, and clusters for surface gather points are represented by cuts in the tree. Multidimensional lightcuts [Walter et al. 2006] is an extension of lightcuts to support efficient rendering of motion blur and depth of field effects. This approach maintains an additional binary tree for clustering gather points. Bidirectional lightcuts [Walter et al. 2012] is the recent extension of lightcuts that combines VPLs with bidirectional subpath tracing and multiple importance sampling in order to render glossy appearance, translucency, and volumetric materials such as cloth. In contrast to instant radiosity, eye subpaths of a few bounces are traced before connecting to the VPLs.

Our goal in this work is to explore how VPLs can be used for directional importance sampling. While a similar idea has been explored before with photons [Jensen 1995; Hey and Purgathofer 2002; Vorba et al. 2014], we further make it more scalable by utilizing clustering. This allows a more general use of VPLs so that it can be integrated to existing Monte Carlo algorithms such as path tracing and bidirectional path tracing. Such algorithms are very general, have been well understood, and can support a wide range of effects including glossy appearance and caustics. A common importance sampling technique that is often used in Monte Carlo path tracing is BRDF sampling. While this is a robust technique, it does not consider the incoming radiance distribution.

4.1.2 Importance sampling with VPLs

There have been a few approaches that utilize VPLs for importance sampling. Georgiev et al. [2012] proposed to evaluate VPLs at a sparse set of surface points and cache the light distribution for importance sampling. Their goal is to use the cache to find a set of most relevant VPLs for each gather point. This can be regarded as an alternative approach to VPL clustering. Wu and Chuang [2013] proposed to build the incoming radiance, BRDF, and visibility distribution from the VPL clusters. Their goal is also to choose a subset of VPLs based on such distributions for rendering. However, these approaches do not render glossy surfaces effectively. There is no eye subpaths generated at the gather points.

Our method is closely related to the techniques in [Georgiev et al. 2012] and [Wu and Chuang 2013]. However, our goal differs. We aim to build a probability distribution that is proportional to the incoming radiance, and then use it to sample eye subpaths in path tracing. As traditional path tracing is already effective in rendering glossy appearance using BRDF sampling, adding importance sampling of the incoming radiance can further improve its effectiveness in handling smooth diffuse surfaces. Recently, Vorba et al. [2014] also explored this idea. They estimate the probability distribution by fitting a Gaussian mixture model to the incoming photons at a surface point. In contrast, we make use of VPLs so that unoccluded long-range VPLs can also participate to build the probability distribution. Our importance sampling technique is based on Metropolis sampling and hence does not require fitting. The only required operation is the estimation of the incoming radiance for a particular direction. In addition, we make our method scalable by clustering the VPLs and use the cluster representatives to estimate incoming radiance distributions.

Strictly speaking, estimating the incoming radiance using VPLs is a chicken-and-egg problem because many-light rendering would not be able to accurately handle surface appearance such as glossiness and caustics. Therefore, while importance sampling with distribution estimated by VPLs would be imperfect, we show that such a best-effort distribution can still perform effectively and produce low-noise images.

Our method is also related to Metropolis sampling, which is first introduced to physically based rendering in the seminal work by Veach and Guibas [1997]. In Metropolis light transport, a Markov chain of light paths is constructed for the entire image and thus when the chain is in its equilibrium state, its sample resembles a distribution that is proportional to the contribution of light paths. In our approach, we use Metropolis sampling to construct the probability distribution that is proportional to the incoming radiance distribution for a set of gather points in the scene. Such probability distribution can then be utilized for directional importance sampling.

4.2 Our method

Our key idea is to use incoming radiance estimated by VPLs to guide directional importance sampling. An overview of our method is illustrated in Figure 4.1. This section describes the details of our approach.

We begin with the rendering integral at a gather point $\mathbf{x}$, which can be written as

$$L(\omega_o) = \int_\Omega L(\omega_i) f_s(\omega_i, \omega_o) \cos(\omega_i, \mathbf{n}) \mathrm{d}\omega_i, \tag{4.1}$$

where $L(\omega_o)$ and $L(\omega_i)$ are the radiance outgoing and incident at the gather point, $f_s(\omega_i, \omega_o)$ the BRDF at the gather point, and $\mathbf{n}$ is the surface normal, Ω the unit hemisphere domain.

This integral is estimated by summing the contribution from a set of VPLs:

$$\begin{aligned} L(\omega_o) = \sum_k & \Phi_k G_k V_k \\ & \cdot f_s(\omega_{in}(\mathbf{y}_k) \to \mathbf{y}_k \to \mathbf{x}) f_s(\mathbf{y}_k \to \mathbf{x} \to \omega_o), \end{aligned} \tag{4.2}$$

where $\mathbf{x}, \mathbf{y}$ are the locations of the gather point and the VPL, respectively, k the index of the VPL, Φ the power of the VPL, G the form factor, V the visibility between the gather point and the VPL, $f_s(\omega_{in}(\mathbf{y}_k) \to \mathbf{y}_k \to \mathbf{x})$ and $f_s(\mathbf{y}_k \to \mathbf{x} \to \omega_o)$ the BRDF at the VPL and the gather point, $\omega_{in}(\mathbf{y}_k)$ and ω_o the incoming direction at the VPL and the outgoing direction at the gather point, respectively. Since each VPL defines an incoming direction for the gather point, we can rewrite the equation such that the sum is over a set of directions:

$$L(\omega_o) = \sum_{\omega_i} I(\omega_i) f_s(\omega_i, \omega_o), \tag{4.3}$$

where ω_i corresponds to the incoming direction by each VPL k. We define the incoming radiance from a VPL to the gather point as

$$I(\omega_i) = \Phi_k G_k V_k f_s(\omega_{in}(\mathbf{y}_k) \to \mathbf{y}_k \to -\omega_i), \tag{4.4}$$

where $-\omega_i$ denotes that the incoming direction at the gather point is the outgoing direction at the VPL.

From the above equations, we see that $I(\omega_i)$ is equivalent to $L(\omega_i) \cos(\omega_i, \mathbf{n})$ in the rendering integral. In other words, the VPLs define the incoming radiance distribution. Our goal is to use function $I(\omega)$ for importance sampling. In particular, we estimate the outgoing radiance using Monte Carlo integration:

$$L(\omega_o) = \frac{1}{n} \sum_{j=1}^{n} \frac{L(\omega_j) \cos(\omega_j, \mathbf{n}) f_s(\omega_j, \omega_o)}{p(\omega_j)}, \tag{4.5}$$

where ω_j is the incoming direction sampled according to the probability distribution $p(\omega)$ which $p(\omega) \propto I(w)$.

To sample $I(\omega)$, we propose to use Metropolis sampling, which is a general technique to sample a function of unknown distribution but evaluation of the function is available. The key idea is to build a Markov chain such that its histogram resembles the function we want to sample.

In order to use Metropolis sampling, it is necessary to evaluate the incoming radiance for an arbitrary gather point. Unfortunately, this evaluation is not available as the incoming radiance is in fact what we are trying to estimate from the rendering equation. However, we can approximate the incoming radiance distribution using the VPLs, i.e., the distribution $I(\omega)$. Therefore, it is possible to apply Metropolis sampling as long as we are able to evaluate $I(\omega)$ for an arbitrary direction ω.

4.2.1 Estimating incoming radiance

For each gather point, the VPLs represent a discrete incident light field over a fixed set of directions. In order to use this distribution for sampling, it is necessary to estimate the incoming radiance $I(\omega)$ for any arbitrary direction. There have been a few studies regarding this problem. Jensen [1995] proposed to construct a 2D map at the gather point which records the incoming radiance that falls into each map cell. Sampling process boils down to importance sampling a cell, and then uniformly select a direction in the cell. Vorba et al. [2014] proposed to estimate the light field using Gaussian mixture model and EM optimization. In this work, we choose to not use fitting as it requires selecting a model and an additional optimization. We assume the incoming radiance distribution can be fit into the memory so that parameterization of the distribution is not necessary.

At a gather point, we use the following approach to estimate the incoming radiance $I(\omega)$, which is very similar to the approach by Hey and Purgathofer [2002]. We first construct a cone at the gather point which is centered towards direction ω. We then query the VPLs that fall into the cone and are visible to the gather point. The incoming radiance is estimated by averaging the contributions from these VPLs. For convenience, we perform such computations on a 2D domain by mapping the unit hemisphere to the unit square $[0, 1]^2$. We note that this domain is continuous and we are not limited by the resolution as when estimating incoming radiance in a grid [Jensen 1995].

4.2.2 Metropolis sampling

After being able to evaluate $I(\omega)$, we are now ready to apply the Metropolis algorithm to sample it. Metropolis sampling is a robust and very general importance sampling technique;

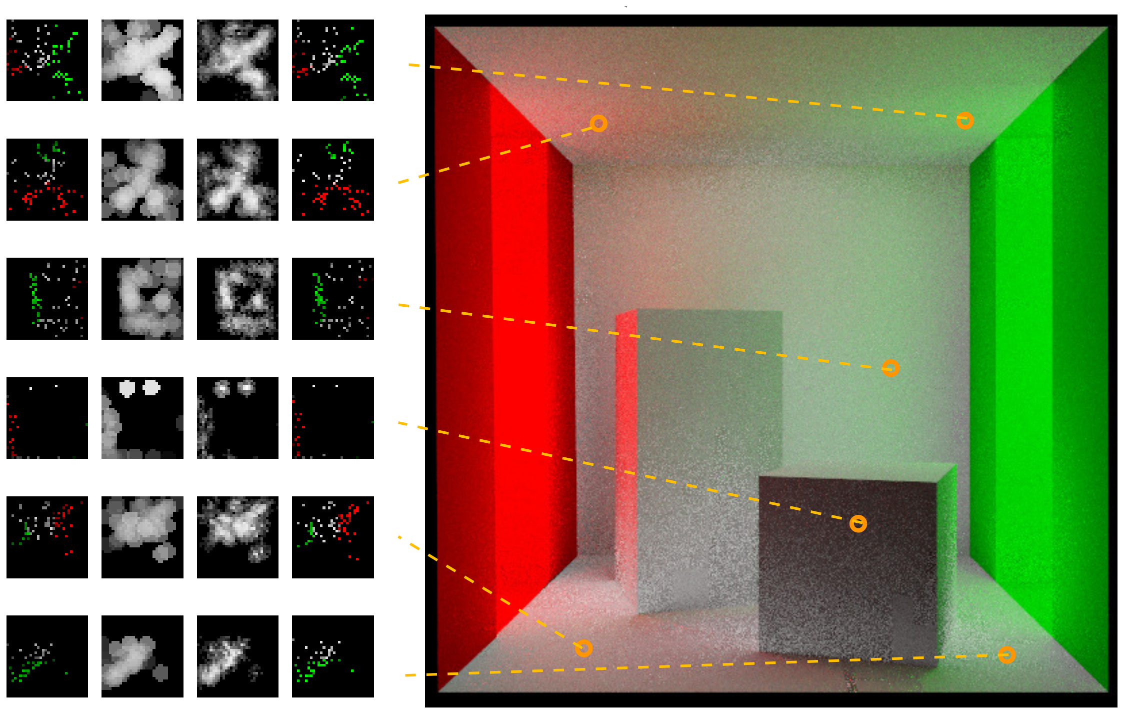


Figure 4.2: Visualization of incoming radiance distributions at various points in the Cornell box scene, from left to right: (i) Incoming radiance as seen from the nearest cache point; (ii) The density map; (iii) Histogram from the Metropolis sampler; (iv) Ground truth incoming radiance seen from the gather point.

it can be used to sample a distribution $f(x)$ where f is unknown but only its evaluation is available, which is exactly our case.

Metropolis sampling draws sample from a Markov chain. However, it is impractical to build a Markov chain at each gather point because there are millions of gather points and the chance that a gather point is revisited so that its Markov chain can be reused is zero in path tracing. However, it can be observed that the incoming radiance distribution at a gather point is very similar to that of other gather points in its local neighborhood. Therefore, we propose to only build Markov chains for a small set of gather points and cache them. The Markov chains at these cache points can be reused for sampling at other gather points. At each cache point, we build a Metropolis sampler to explore the space of directions based on its incoming radiance distribution. At each gather point, the nearest cache point is queried and its Markov chain can be used to sample a new direction for the gather point. Note that our Metropolis sampling is therefore different from Metropolis light transport [Veach and Guibas 1997], in which a single Markov chain is run for the entire image. In our case, a Markov chain is stored at each cache point.

Figure 4.2 depicts the incoming radiance distribution at various gather points in the Cornell

Algorithm 4.1: The Metropolis algorithm to sample new directions and fill the sample buffer. The current direction in the Markov chain is ω.

```
1  while the sample buffer is not full do
2      Select a mutation type $m$ with probability $p(m)$.
3      Sample a new direction $\omega'$ using the transition probability $T_m(\omega' \mid \omega)$.
4      Compute $T(\omega' \mid w) = \sum_m p(m) T_m(\omega' \mid \omega)$. Similarly, compute $T(\omega \mid \omega')$.
5      Let $a = \min\left(\frac{I(\omega')T(\omega \mid \omega')}{I(\omega)T(\omega' \mid \omega)}, 1\right)$.
6      if $rand() < a$ then
7          Accept proposal $\omega = \omega'$.
8      else
9          Revert $\omega' = \omega$.
10     end
11     Compute probability $p(\omega') = I(\omega')/b$.
12     Store tuple $(\omega', p(\omega'))$ to the sample buffer.
13 end
```

box scene. As can be seen, the incoming radiance distribution at the gather points and their nearest cache points are very similar, and the Metropolis algorithm is able to produce samples that closely follow such distributions.

The core of a Metropolis sampler is a set of mutation techniques that propose samples for the Markov chain. Recall that a new state in a Markov chain only depends on its current state. We suggest using two mutations: uniform sampling a new direction and perturbing the current direction in the Markov chain about a small cone. Such mutations are general and allows ergodicity. The first mutation attempts to explore the entire unit hemisphere, where the new sample is independent of the current sample in the Markov chain. The second mutation tries to explore the local neighborhood of the current sample. These mutations are symmetric, and thus the transition probabilities in the Metropolis algorithm cancel out.

At each cache point, we store the samples generated by the Metropolis sampler into a sample buffer. This sample buffer is used to avoid correlation. The order of samples are scrambled by a permutation every time the buffer is filled by the Metropolis algorithm. This ensures that two gather points that are very close to each other and has the same nearest cache point do not use correlated directions. Our Metropolis sampling algorithm to generate a new direction ω' from a current direction ω in the Markov chain is listed in Algorithm 4.1.

4.2.3 Estimating the total incoming radiance

In order to sample with $I(w)$, we need to estimate the density $p(\omega) = I(\omega)/b$, where $b = \int I(\omega)\mathrm{d}\omega$ is the normalization factor to transform $I(\omega)$ into a probability distribution. This value can be easily approximated by summing the splats of the incoming radiance from all VPLs:

$$b \approx 2\pi(\pi r^2) \sum_k I_k \tag{4.6}$$

where I_k is the incoming radiance from VPL k, r the radius of the splat disk on the unit square. The constant 2π accounts for the fact that the integration over the unit hemisphere is performed on the unit square domain.

There might have a few directions that have no nearby VPLs, and thus the incoming radiance estimation results in zero. However, to make the sampler unbiased, it is necessary to explore such directions. To solve this, we add back a small ratio of total incoming radiance β each time the incoming radiance is estimated. We set the ratio to 1%.

4.2.4 Sampling the product of incoming radiance and BRDF

A light path can now be constructed incrementally using more than one technique, BRDF sampling or our Metropolis sampler. BRDF sampling is robust for glossy surfaces, while our sampler is more effective when the BRDF at gather points are more diffuse and the incoming radiance contains high frequencies. Therefore, it could be best to combine these samplers together. There are a few possibilities to perform this task. An option is to generate two samples, each by sampling $I(\omega)$ and $f_s(\omega)$, and then combine them using weights computed by balance heuristics in multiple importance sampling (MIS) [Veach 1998].

Another option is to use resampled importance sampling (RIS) [Talbot et al. 2005]. In order to sample $I(\omega)f_s(\omega)$, a set of M directions ω such that $p_1(\omega) \propto f_s(\omega)$ is sampled. $I(\omega)$ for all ω in the set is then evaluated to build a discrete probability distribution $p_2(\omega) \propto I(\omega)$. By using p_2 to sample a direction in the set, the final probability of the sample would be $p(\omega) = Mp_1(\omega)p_2(\omega)$ and thus proportional to the product $I(\omega)f_s(\omega)$. For variance analysis of RIS, please see [Talbot et al. 2005]. A few previous works had followed this approach. Burke et al. [2005] estimate radiance due to an environment light by sampling the product of the BRDF at the gather point and the irradiance distribution. Wang and Åkerlund [2009] extend this idea by discretizing the environment map into VPLs. They estimate the average BRDF for each VPL cluster and then use the product of average BRDF and cluster power to sample clusters for visibility test. Georgiev et al. [2012] use radiance without visibility test to resample a subset of VPLs to gather.

Since the number of samples in a set in RIS can be quite limited, we choose to sample $I(\omega)$ and $f_s(\omega)$ separately and combine their contributions using MIS. To our experience, this is often more flexible than RIS as we can use as many samples as possible.

4.2.5 VPL clustering

Since evaluating every VPL at each cache point is expensive and there can be up to millions of VPLs, we advocate the use of VPL clustering. Only cluster representatives are considered to build incoming radiance distribution. The incoming radiance from a cluster representative is scaled by the weight of the cluster. We choose to use LightSlice [Ou and Pellacini 2011] as our clustering technique. The steps to build clusters are:

1. Generate light subpaths and cache each path vertex as a VPL.
2. Generate eye subpaths and gather points visible to the camera.
3. Cluster gather points into slices and select slice representatives.
4. Cluster the VPLs for each slice.

We assign each slice representative as a cache point and thus build a Metropolis sampler for it. By following [Ou and Pellacini 2011], we use the term *slice* to refer to a group of gather points and the term *cluster* to refer to a group of VPLs. We also refer to points that are slice representatives as cache points.

We use matrix row-column sampling [Hašan et al. 2007] to construct a global clustering of VPLs for all the slices. The clusters per slice are then refined in a top-down manner by first projecting the columns onto a hyperdimensional line and then splitting the line into two. The only difference is that each entry in the matrix represents incoming radiance $I(\omega)$ from each VPL to each slice representative instead of outgoing radiance as in [Hašan et al. 2007].

4.3 Implementation details

We implement our Metropolis sampling approach on top of LightSlice clustering, and experiment it with a two-bounce path tracer, i.e., the max path length is three. In our current implementation, cache points are chosen only from gather points that are visible to the camera. In particular, the visible gather points are clustered using a 6D-kdtree as in [Walter et al. 2006] and their representatives are marked as cache points. We note that it is possible to use our sampler for exploring deeper bounces of the light transport. This can be done by adding cache points not just at visibe gather points, and also incrementally placing additional cache points when there is no nearby cache points as in [Vorba et al. 2014].

We use both BRDF sampling and Metropolis sampling, and combine the contributions using MIS. To avoid exponential generation of light paths, either BRDF sampling or Metropolis sampling is chosen randomly as the technique to use every time a new direction is needed.

At each cache point, we build a kd-tree for 2D points that are mapped to the unit square from cluster representatives. The incoming radiance for an arbitrary point is then estimated by considering incoming radiance samples from its neighbors within a radius in the unit square. This radius controls how smooth the probability distribution is. Increasing the radius makes the incoming radiace more uniform.

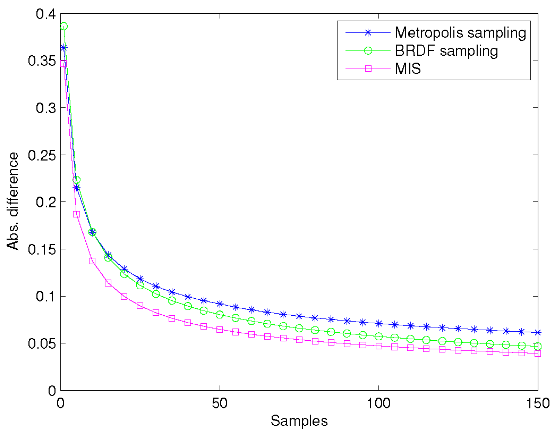
Correlation can occur at consecutive gather points that use the same cache point to generate new directions. Artifacts due to correlation depend on the order of the gather points to render. For example, if the pixels are scanned line by line, from left to right, correlation artifacts can appear as horizontal lines. To reduce such artifacts from early iterations, it is necessary to let the pixel order be independent of the sampling process. To achieve this, we store a permuted set of samples generated by the Metropolis sampler into the sample buffer at each cache point. The permutation order is regenerated every time the sample buffer is refilled. This permutation hides the correlation artifacts while still correctly maintaining the Markov chain at each cache point. We set the sample buffer size to 1024 samples.

To sample a new direction at a gather point, its nearest cache point is queried. This can be done by building a kd-tree that contains all cache points. We use the 6D kd-tree as in [Walter et al. 2006; Ou and Pellacini 2011] which considers both location and orientation of the cache points. To make sure that the incoming radiance distribution at the cache point and the gather point are sufficiently close, we reject cache points that have dissimilar orientations from that at the gather point. When a valid cache point is found, its Metropolis sampler is then used to draw a new direction. We assume that in the local coordinate frame established by the surface normal and tangent, the new direction at the cache point and the gather point is the same. If no cache point is found similar to the gather point, the sampling fails and a zero direction should be returned with a zero probability density value. However, this can sometimes lead to black patches in the result. To avoid this, we instead revert to uniform sampling and return the corresponding probability.

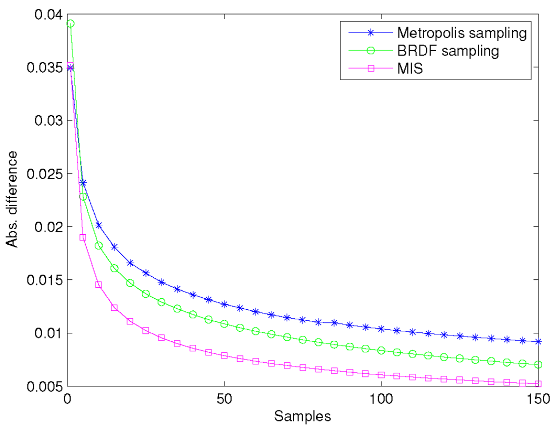
We build Metropolis sampler only for opaque surfaces. Therefore, only cluster representatives that are unoccluded to the cache point and have positive form factors, i.e., positive cosines at the VPL and the cache point, are used for incoming radiance estimation and directional sampling.

4.4 Experimental results

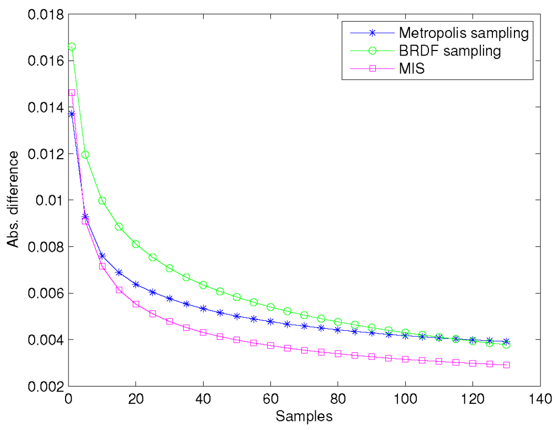
We implemented our method in C++. All time measurements are done on a machine with an Intel Core i7-4770 CPU clocked at 3.40 GHz and 12 GB of RAM. Our path tracer is single threaded. All images are rendered in 1024×768 with 2×2 supersampling and box reconstruction filter. We only use our sampler to guide indirect illumination, and hence only indirect illumination is included in the results. All our experiments are done with a



(a) Kitchen



(b) Breakfast



(c) Conference

Figure 4.3: Absolute error plots of the example scenes. While Metropolis sampling does not always outperform BRDF sampling, combining both of the techniques using MIS gives far more accurate results.

two-bounce path tracing. To ensure fair comparison, only VPLs that bounce once are used as incoming radiance samples so that the maximum path length is three. We fixed the number of cache points to 1024 in all our examples.

All mutations in the Metropolis sampler are set to be chosen with equal probability. The cone size in the perturbation mutation technique is set to $\pi/20$. Using a too small cone size can cause the directional space not well explored.

We render three scenes with complex illumination: the Kitchen scene adapted from [Hardy 2012], the Breakfast scene [Wayne 2014], and the Conference scene [McGuire 2011]. The scenes contain occluded light sources and a mix of diffuse and glossy surfaces.

We compare the results generated by Metropolis sampling, BRDF sampling, and MIS which combines both of these sampling techniques. The ground truths are generated using BRDF sampling with a large number of paths per pixel. The plots in Figure 4.3 demonstrate the absolute difference of the images produced by Metropolis sampling, BRDF sampling, and MIS with the ground truths. It can be seen that Metropolis sampling is able to converge, and overall its performance is comparable to BRDF sampling. The Metropolis sampler tends to be more effective in smooth regions where BRDF sampling does not work well. On the other hand, the Metropolis sampler works less effectively in glossy regions. Therefore, the MIS of Metropolis sampling and BRDF sampling yields the best results. This can also be validated by examining the error heat maps in Figure 4.4.

The MIS results are generated as follows. We randomly select between Metropolis sampling and BRDF sampling with probability 0.5 when a direction is needed and combine the results using balance heuristics [Veach 1998]. To ensure a fair comparison, the MIS image is generated with the same number of samples as used in Metropolis and BRDF sampling. Each sample can be either a Metropolis sample or a BRDF sample.

Figure 4.4 shows the rendered images, error heat maps, and the ground truth images. The error maps depict the relative error of the Metropolis sampling, BRDF sampling, and MIS with the ground truth, respectively. The error in $[0, 1]$ is mapped into color transition from blue to red. Again, it can be seen that Metropolis sampling renders the diffuse and low-gloss surfaces in the scene more effectively thanks to the importance sampling of incoming radiance. In contrast, BRDF sampling works more effectively in glossy regions. Therefore, the results of MIS that combines both of the techniques have the lowest errors.

We also render our scenes using the implementation in Mitsuba [Jakob 2010] provided by Vorba et al. [2014]. The results in Figure 4.4 show that our MIS works as well as their approach, which therefore proves the effectiveness of the Metropolis sampler.

As can be seen, the Metropolis sampler can causes some visual artifacts. This is because the Metropolis sampler fails when the local geometry orientation at a gather point changes abruptly and similar cache points cannot be found. In such cases, uniform sampling is used for such gather points, which leads to slower convergence. The convergence speed difference among gather points thus appear as visual artifacts. However, we note that it is still possible for such gather points to convergence. In most of the cases, MIS can lower the weight of Metropolis sampling at locations where artifact occurs, and thus still produces high-quality images.

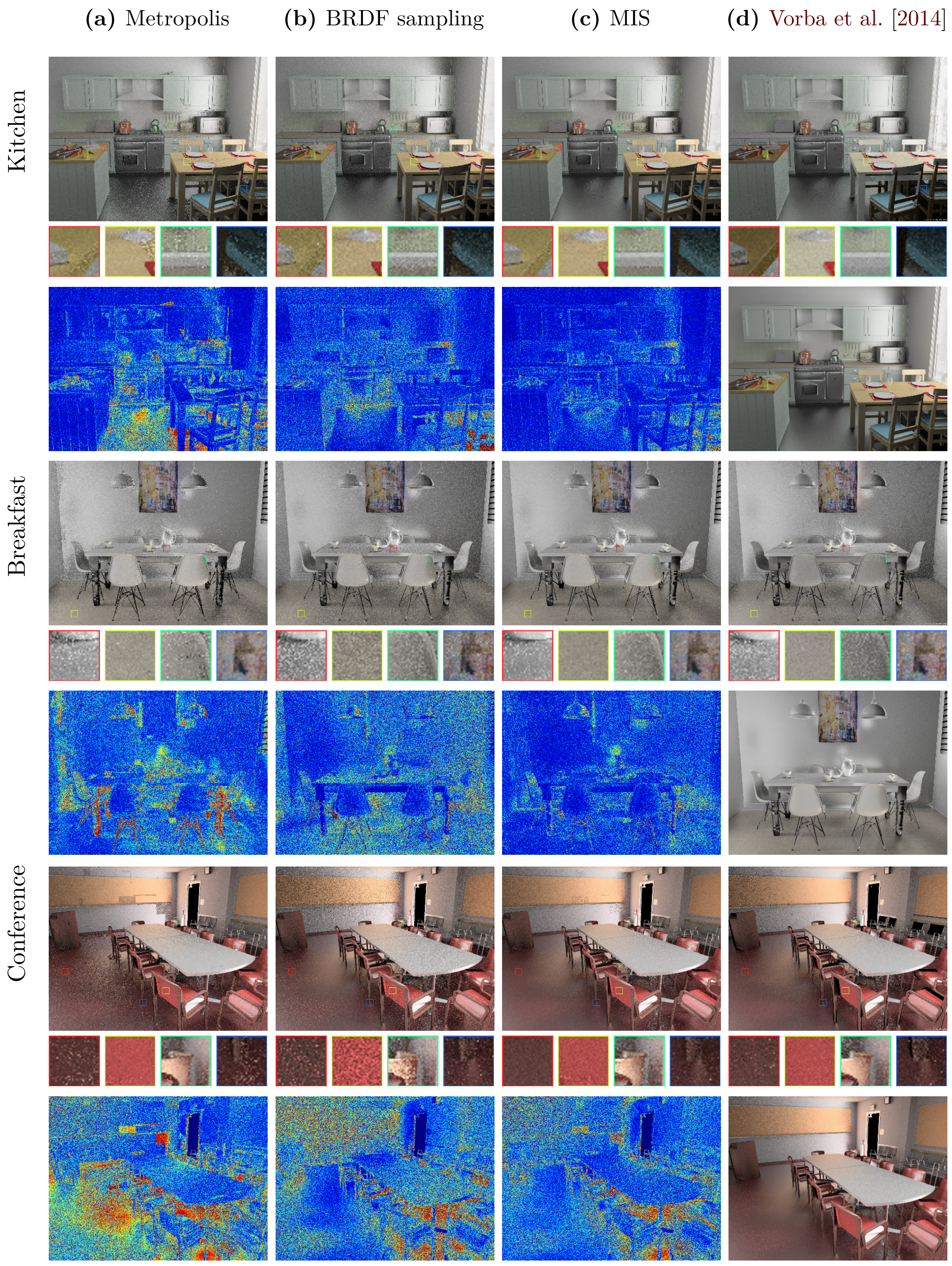


Figure 4.4: The results of our tested scenes. Odd rows: results by Metropolis sampling, BRDF sampling, MIS, and by Vorba et al. [2014]. Even rows: error heat map of Metropolis sampling, BRDF sampling, MIS, and the ground truth.

	Kitchen	Breakfast	Conference
VPLs	50 K	48 K	39 K
VPL clusters	1200	2000	1200
Paths	150	150	130
Initialization	7.5 mins	8.25 mins	3 mins
Path tracing	311 mins	280 mins	233 mins
Memory	2.6 GB	1.5 GB	1.5 GB

Table 4.1: Statistics of our scenes rendered using MIS.

We report the statistics of our results rendered using MIS in Table 4.1. The running time of each scene has two parts: initialization that includes VPL clustering using LightSlice, and path tracing up to the specified number of paths per pixel. As our implementation is not yet optimized, the reported running time and memory usage can be further improved.

4.5 Conclusions

In this chapter, we proposed a new importance sampling approach that utilizes the incoming radiance distributions estimated by VPLs. We demonstrated that our method works effectively and can be easily integrated into path tracing and LightSlice.

There are a few limitations in this work. It can be observed that our Metropolis sampler can be inefficient if the technique used to estimate incoming radiance provides a poor approximation. At gather points that use such a distribution, although it is possible to converge eventually, the convergence is far more slower and thus causes artifacts in the results. In a few cases, MIS with balance heuristics might not be able to effectively hide such artifacts. We plan to explore how to combine the Metropolis sampling with BRDF sampling more effectively. In addition, so far our sampler had only been used to guide unidirectional path tracing. We aim to apply it to bidirectional path tracing for greater efficiency.

Chapter 5

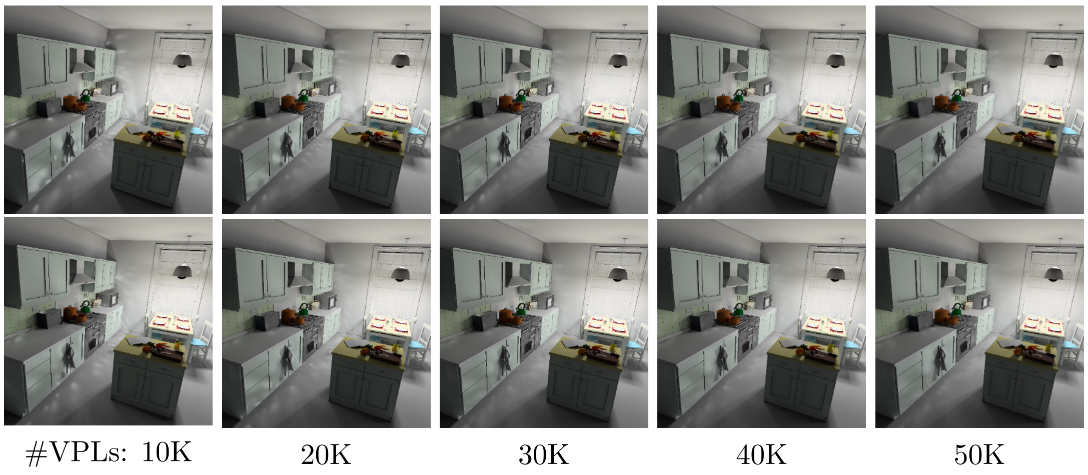
Reducing artifacts in many-light rendering

Many-light rendering approximates indirect illumination by a large set of virtual point lights. The outgoing radiance at a surface point is the total contribution from the point lights, each weighted by the BRDF and geometry terms. Evaluating outgoing radiance can be efficiently implemented on the GPU using shadow mapping.

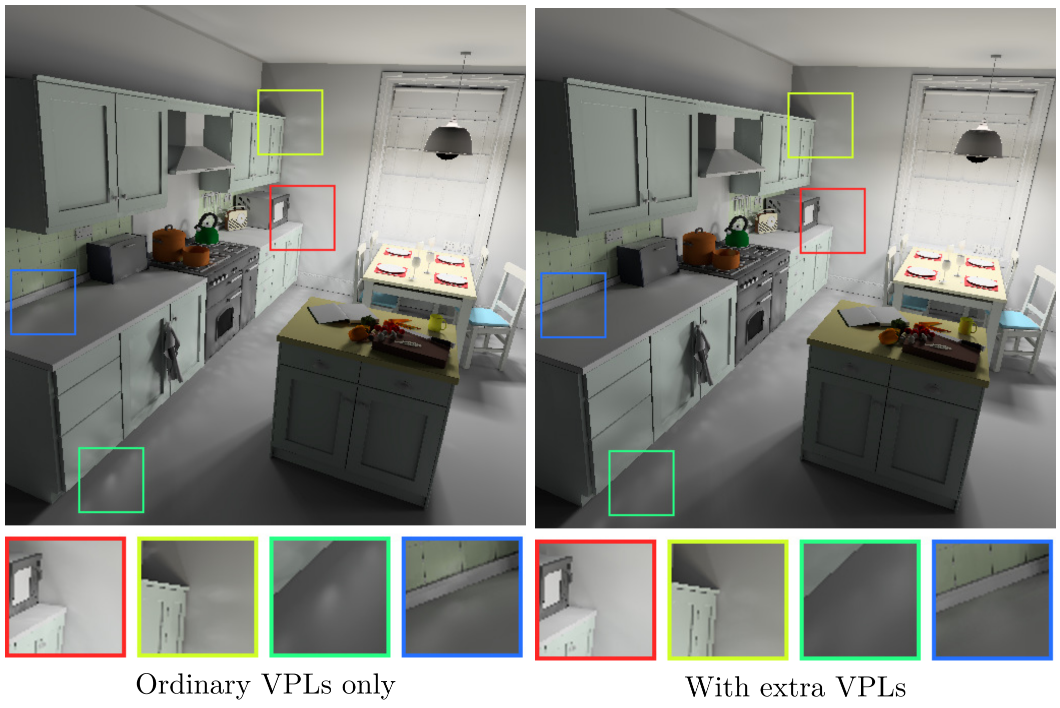
Unfortunately, the gathering step has been known to introduce singularity, which is due to that fact that the distance from a VPL to a receiver can approach zero. When this happens, the geometry term and the outgoing radiance can become infinite. The final image thus can exhibit very bright spots that are often located close to edge and corner regions of the scene. For glossy surfaces in the scene, bright spots can also occur when the BRDF lobe of the VPL aligns to the BRDF lobe of the receiver. Bright spots also appear when the density of VPLs in a local neighborhood is too low, and the contribution of a VPL to a surface becomes easily identified. Some of such bright spots cannot simply be addressed by accumulating more VPLs as the number of VPLs required can be too large.

The common technique to address this problem is by limiting the overall radiance contribution of the VPL to the receiver, which is known as *clamping*. However, clamping can further cause dark corners, where a significant amount of illumination from the gathered VPLs is lost. When rendering glossy surfaces, clamping also leads to loss of specular highlights and make glossy surfaces appear to be diffuse. Therefore, in general it is more desirable to set the clamping threshold as high as possible to mitigate energy loss.

In this chapter, we propose an adaptive VPL sampling method that aims to reduce bright spot artifacts as early as possible in progressive rendering. Our VPL sampling method can be paired with standard VPL gathering with minor clamping to reduce sharp bright spots. We demonstrate a progressive multi-pass rendering framework in which VPLs of a latter pass are generated based on feedback from rendering results of the former pass. We show that our method can reduce the bright spots and achieve comparable convergence rate to traditional VPL gathering.



(a) Progressive rendering. First row: ordinary VPLs only. Second row: with extra VPLs.



(b) The scene rendered with 60K VPLs.

Figure 5.1: Progressive rendering of the Kitchen scene [Hardy 2012]. Our method allows progressive rendering with less bright spots.

5.1 Related works

Several prior works have been proposed to address the bright spot and dark corner issue in VPL gathering. Generally, these can be categorized as two classes of approaches, clamping-free gathering methods, and bias compensation.

Clamping-free VPL gathering The central idea of clamping-free methods is to perform VPL gathering in a way such that weak singularity does not appear. One of the first work of this kind is virtual spherical light (VSL) [Hašan et al. 2009], which is based on the concept of photon mapping and photon light. Each VSL is a virtual light of which illumination is dispatched to a surface region confined in a sphere. Evaluating a VSL requires to sample rays in the cone subtended by the sphere, in which the weak singularity due to inverse squared distance is avoided. This approach works very well, and can support glossy surfaces, but it cannot handle highly detailed glossy surfaces due to the splatting of incident illumination. It also requires that visibility of all rays in the cone is approximately the same. This allows VSL to fit into the traditional shader pipeline and to use shadow mapping for visibility check between a VSL and all receivers. Virtual ray light (VRL) and virtual beam light (VBL) are variants of VSL for rendering participating media.

Our VPL sampling approach is closely related to VSL. The new VPLs that are generated is also from a cone sampling. However, this strategy is only used for exploring the local regions around a VPL. The gathering step in our framework follows a standard VPL evaluation.

Bias compensation In contrast to clamping-free methods, bias compensation approaches allow clamping but then calculate and add back the amount of missing energy due to clamping. Kollig and Keller [2006] formulated the missing energy as a rendering integral which has no singularity, and solved it by perform an extra path tracing step after VPL gathering. In principle, this method works well, but in practice, performing path tracing to compute the bias can be even more costly than gathering the VPLs. Davidovič et al. [2010] proposed to use local lights to perform compensation and render glossy surfaces at the same time. The idea is to split the rendering into two steps which separately evaluates the global low-rank light transport with VSL, and the local high-rank light transport with local VPLs generated from pixel tiles. Even though their method can render detailed glossy surfaces, the visibility of local VPLs are ignored. Our VPL sampling is rather similar to the local light generation. However, we aim to discard artifacts in the VPL gathering as early as possible.

In an earlier work, Křivánek et al. [2010] proposed to scale the clamped illumination so that it matches the average illumination of the no-clamped image. However, this requires to obtain an estimate of the average illumination of the scene. Another natural way to reduce bias is to introduce more VPLs. This increases the density of VPLs in local regions and reduces the power each VPL conveys, thus reduces clamping. Walter et al. [2012] reduces bias by introducing virtual sensor points (VPS) from sampling eye subpaths. They proposed to connect VPLs and VPSes using multiple importance sampling with an adaptive set of weights. They pointed out that the bias in clamping can be expressed as constraints on path weights that can sum to less than one, and proposed to implement clamping as one of their four weight constraints. These methods are orthogonal to ours, and we focus on discarding rendering artifacts by performing adaptive VPL sampling.

VPL sampling Conventionally, generating VPL is usually done by tracing light subpaths and storing a VPL at each subpath vertex. Segovia et al. [2006] showed that VPL can also be generated by tracing eye subpaths. Georgiev and Slusallek [2010] proposed a simple Russian Roulette approach to find VPLs that can contribute significantly to the camera view. While such approaches sample VPLs, they do not explicitly address artifacts, but rather focus on importance sampling so that the VPL can contribute best to the final image. Their sampling process is independent of the gathering process and there is no explicit mechanism to guarantee that artifacts do not occur in the final result. To the best of our knowledge, our method is the first that attempts to discard artifacts as early as possible in the gathering. We achieve this by explicitly shooting more VPLs into problematic scene regions that can be easily identified after each standard VPL gathering pass.

5.2 Virtual point light

Traditionally, VPLs are generated by sampling light subpaths, which we refer to as standard VPL sampling. VPL evaluation with clamping is considered as standard VPL gathering.

The outgoing radiance at a receiver at $\mathbf{y}$ illuminated by a VPL at $\mathbf{p}$ can be written as

$$L(\mathbf{y}, \omega_o) = \Phi f_s(\omega_i, \mathbf{p}, \mathbf{y}) G(\mathbf{p}, \mathbf{y}) f_s(\mathbf{p}, \mathbf{y}, \omega_o), \tag{5.1}$$

where Φ is the power of the VPL, f_s the bidirectional reflectance function (BRDF), G the form factor which is the product of the geometry term and two cosines, ω_i the direction of incident radiance to the VPL, and ω_o the direction of outgoing radiance at the receiver.

Illumination spikes occur due to the singularity in the form factor G when the distance between $\mathbf{p}$ and $\mathbf{y}$ is very small, and due to large numerical value of the product of the BRDF at $\mathbf{p}$ and $\mathbf{y}$. Such spikes make the radiance L to be a large numerical value and hence appear as bright spots in the final image.

To discard artifacts, we bound the reflectivity of each VPL to a gather point using a maximum threshold τ:

$$L_c(\mathbf{y}, \omega_o) = \Phi \max\left(\tau, f_s(\omega_i, \mathbf{p}, \mathbf{y}) G(\mathbf{p}, \mathbf{y}) f_s(\mathbf{p}, \mathbf{y}, \omega_o)\right), \tag{5.2}$$

where the threshold τ can be set by user to control how quickly the bright spots can fade out in the gathering. A too low threshold will cause glossy surfaces to appear to be completely diffuse. Here we choose to clamp the reflectivity; however, other clamping techniques such as clamping the entire contribution from a VPL as in [Walter et al. 2005] is also applicable.

5.3 Our method

Mathematically, bright spots in instant radiosity are caused by the weak singularity and the BRDFs in Equation 5.1. Dark corners then follows due to energy lost in clamping in Equation 5.2. However, from the perspective of lighting design, another important reason is that the density of lights is too low. When a smooth and large region is illuminated by only a small bright light, it becomes easier to identify the location and orientation of the light. In Monte Carlo path tracing, such bright spots also appear, but in the form of single bright pixels.

This observation leads to our adaptive VPL sampling method. It is desirable to generate VPLs densely in local neighborhood so that their contribution can blend together and no contribution from individual VPLs can be identified in the final image. This is the core of our approach to reduce bright spots.

Our framework follows multi-pass rendering. In each pass, VPLs are first generated and then gathered. After the first pass, VPLs for the next pass are generated by considering the result of the gather step in the previous pass. We minimize changes to the standard VPL gathering process by only adding to it a new functionality: output an additional image that stores the loss energy ratio, which we call the *clamping map*. This map is then used for generating extra VPLs, which aims to reduce artifacts that exist in the previous VPL gathering.

The overall process is as follows. After the first VPL set is generated from sampling light subpaths:

1. Perform gathering. Clamping can be used to discard artifacts. Output the clamping map which marks screen-space pixels where clamping occurs.
2. Scan the clamping map and detect pixel clusters. For each cluster, detect which VPLs actually cause clamping.
3. For each cluster and clamped VPL pair, generate extra VPLs.
4. Repeat.

In the next sections, we discuss these steps in details.

5.3.1 Generating the clamping map

We are interested in knowing if clamping occurs at each receiver. This information can be acquired by querying the clamping map output from the VPL gathering. For each VPL,

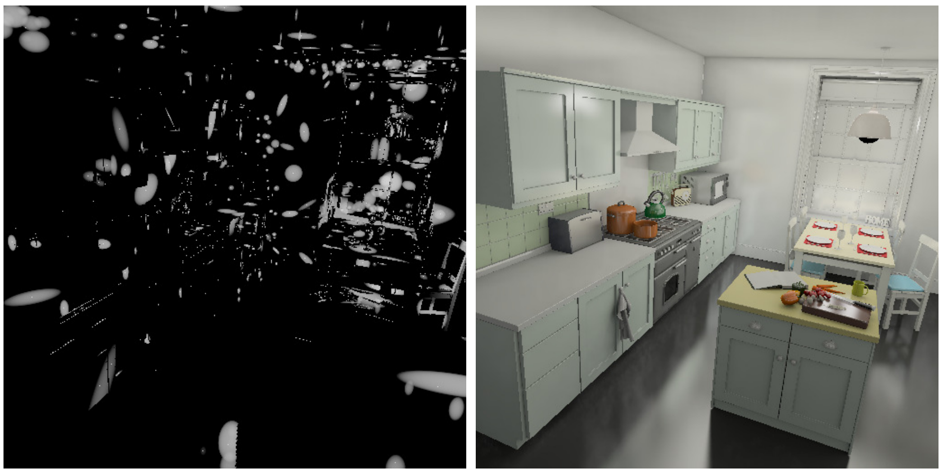


Figure 5.2: A clamping map from the Kitchen scene.

each pixel in the clamping map stores a ratio in $[0, 1]$ which describes how much energy is lost due to clamping. This ratio is calculated as

$$\delta = 1 - \frac{L_c}{L}. \tag{5.3}$$

When no clamping occurs, the ratio is equal to zero. It reaches one when a large amount of energy is clamped.

Since a set of VPLs are evaluated in each rendering pass, the clamping map stores the accumulated ratio for all VPLs in the set.

5.3.2 Analyzing the clamping map

The clamping map records an important piece of information: whether clamping occurs at a receiver and if so, how severe the clamping is. We make use of this feedback to generate VPLs for the next rendering pass, which can compensate and discard the artifacts that might have appeared in the previous rendering pass.

In the clamping map, it can be seen that artifacts are very local in image space. Therefore, we cluster the pixels in the clamping map. Each cluster is a group of pixels that are next to each other and the energy lost ratio is greater than zero. This can easily be achieved by a flood-fill algorithm. For each cluster, the pixel locations and the clamping ratios are stored. They are used to sample pixels in the cluster in the next step.

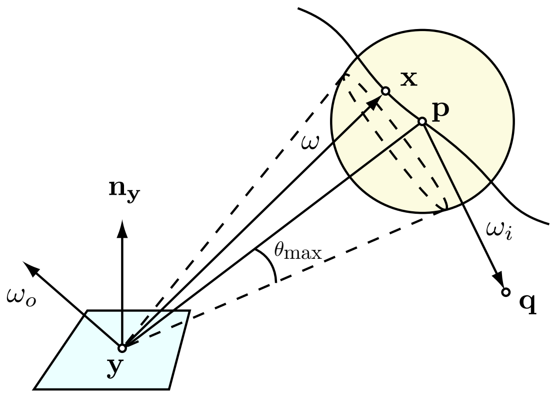


Figure 5.3: Extra VPLs are generated by sampling the cone subtended by a virtual sphere at the VPL that causes artifacts.

5.3.3 Generating extra VPLs

The extra VPLs are generated from length-2 eye subpaths that start from the camera lens and travel through pixels in each cluster. In this section, we refer the VPLs in the previous rendering pass as the ordinary VPLs.

For each pair of pixel in a cluster and an ordinary VPL, the extra VPLs are generated as follows. The process is demonstrated in Figure 5.3.

1. Generate the eye subpath through the pixel. Let $\mathbf{y}$ be the location of the surface receiver. Let $\mathbf{p}$ be the location of the ordinary VPL.
2. Sample rays from $\mathbf{y}$ in the cone subtended by a sphere of radius r centered at $\mathbf{p}$. Calculate the intersection $\mathbf{x}$.
3. Store the extra VPL at $\mathbf{x}$.

The cone can be uniformly sampled. The ray at the local coordinate frame is $\omega = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$, where θ and ϕ can be generated from two random numbers (ζ_1, ζ_2) from a uniform distribution as

$$\begin{aligned} \theta &= \arccos(1 - \zeta_1(1 - \cos\theta_{\max})), \\ \phi &= 2\pi\zeta_2, \end{aligned} \tag{5.4}$$

where $\sin(\theta_{\max}) = r/\|\mathbf{y} - \mathbf{p}\|$.

We keep track of how many extra VPLs that have been generated for each ordinary VPL. The power of an extra VPL $\mathbf{x}$ can be estimated by dividing the ordinary VPL power to the number of extra VPLs:

$$\Phi(\mathbf{x}) = \Phi(\mathbf{p})/n(\mathbf{p}), \tag{5.5}$$

where $n(\mathbf{p})$ is the number of extra VPLs generated for the ordinary VPL at location $\mathbf{p}$.

The probability of the extra VPL $\mathbf{x}$ can be computed as

$$p(\mathbf{x}) = \int_Y \int_P p(\mathbf{x}|\mathbf{y},\mathbf{p})p(\mathbf{y})p(\mathbf{p})\mathrm{d}\mathbf{y}\mathrm{d}\mathbf{p} \tag{5.6}$$

where $\mathbf{y} \in Y$ and $\mathbf{p} \in P$, which are the set of receivers and ordinary VPLs, $p(\mathbf{y})$ and $p(\mathbf{p})$ are probability of selecting the receiver and the ordinary VPL. By assuming that each extra VPL $\mathbf{x}$ can only be generated from a single pair of receiver and ordinary VPL, the probability of $\mathbf{x}$ can be simplified to

$$p(\mathbf{x}) = p(\mathbf{x}|\mathbf{y},\mathbf{p}) = p(\omega)\frac{\cos(-\omega, \mathbf{n}_\mathbf{x})}{\|\mathbf{x}-\mathbf{y}\|^2}, \tag{5.7}$$

where ω is the new direction sampled in the cone located at y.

We note that this sampling process is both relevant to VPL generated by sampling eye subpaths [Segovia et al. 2006] and virtual spherical light [Hašan et al. 2009]. However, our method is adaptive as we only split an ordinary VPL into a set of extra VPLs when necessary. Our method is also easy to integrate into an existing VPL implementation.

5.3.4 Implementation details

To ensure that all artifacts have an opportunity to be addressed, all clusters are examined. For each cluster, to select the receiver to generate extra VPLs, the pixels in the cluster are importance sampled using the energy lost ratio distribution. For every such receiver, a ray is sampled in the cone subtended by the virtual sphere centered at the ordinary VPL. This process is repeated several times, determined by the number of rays used to sample the cone.

To avoid too many VPLs to be generated at a time, we only examine ordinary VPLs which contribute energy to the chosen pixels in a cluster. We check this by performing gathering between each ordinary VPL and each pixel but without visibility check. Note that since the number of clusters are few, often from tens to a few hundreds, this check can be performed very quickly, even on the CPU. This ensures that we only explore the local regions of those ordinary VPLs that have a high chance that cause clamping for the cluster and the set of extra VPLs are not too large.

The radius of the virtual sphere at each ordinary VPL used to generate the cone is fixed. The radius should not be reduced progressively because it is preferable to keep exploring the local neighborhood around the VPLs no matter how dense the VPLs are. Extra VPLs that fall out of the virtual sphere are still accepted. We note that it is also possible to set the radius adaptively based on the VPL density. However, we used a fixed radius in our implementation as we found that it worked well for our test scenes.

Note that the extra VPLs can cause artifacts again due to the singularity in its geometry term. This process may never stop and extra VPL is generated repeatedly. We opt to avoid this case. In fact, another reason is that the local regions of an extra VPL can have already been explored by other extra VPLs in the same cone sampling batch. Generating more VPLs into such regions is therefore not the highest priority. In our implementation, extra VPLs are tagged so that they are not considered for cone sampling in the next extra VPL generation pass.

We normalize the image brightness by dividing the total contribution to the number of ordinary VPLs used. The extra VPLs are already accounted for in the power splatting from their ordinary VPLs, so they should not be double counted. In fact, it is generally difficult to consider the extra VPLs and the ordinary VPLs in a single Monte Carlo estimator because the extra VPLs are generated using a different sampling technique than that of the ordinary VPLs.

We also note that extra VPLs alone can create bias as they do not explore the entire path space. However, this bias is negligible as long as there are sufficient ordinary VPLs as shown in our experiments.

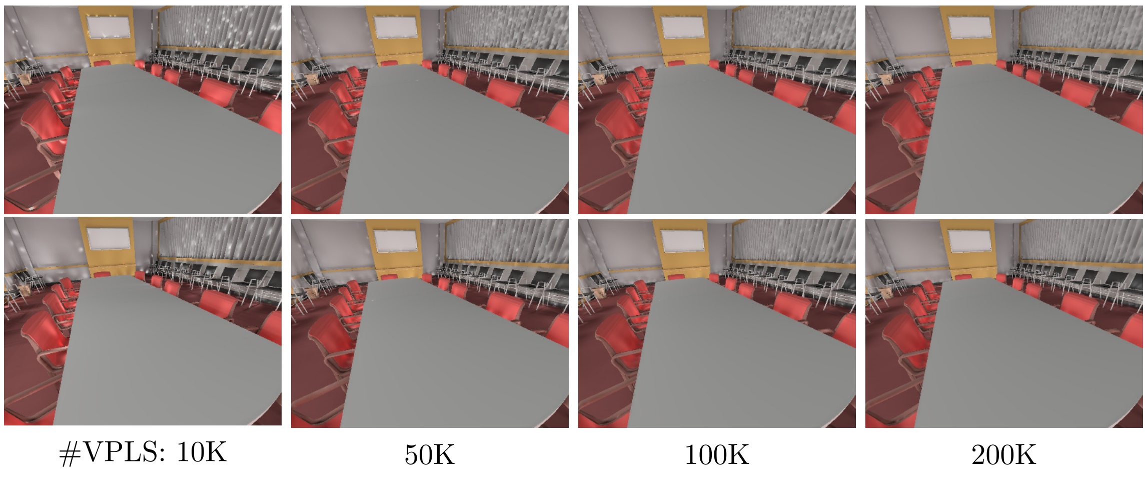
5.4 Experimental results

Our prototype renderer is implemented in C++ and OpenGL 2.1, with VPL gathering and clamping map output implemented in GLSL 1.2 shader. All scenes are presented with only indirect illumination rendered by VPL gathering. Light paths of up to length three are considered. We do not perform VPL clustering in this work. We test the effectiveness of our method with Country Kitchen and the Conference scene. We compare our method with standard VPL approach, which VPLs are generated from light subpaths and eye subpaths. The images of a scene are rendered with the same number of VPLs. Reference images are generated by path tracing to compute the error plots.

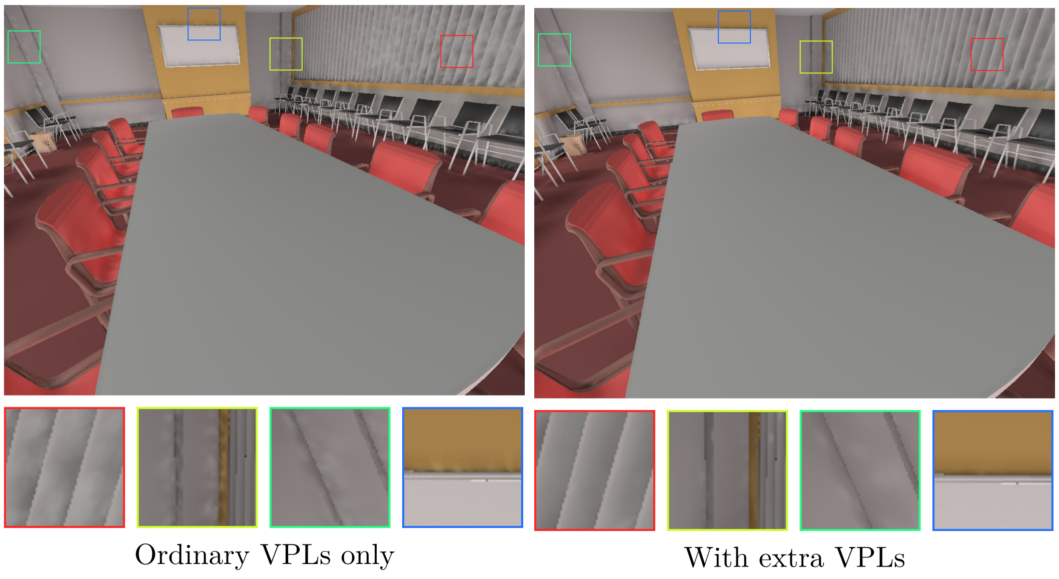
Figure 5.1 shows the rendering of a glossy kitchen scene. We adopted this scene from [Hardy 2012] with BRDF model changed to the modified Phong model. The progressive rendering with total VPLs ranging from 10K to 60K is demonstrated. At the early stages, artifacts tend to be visible when gathering VPLs using the traditional approach, even with clamping applied. In contrast, our method can produce images with less artifacts. The glossy reflection also becomes smoother. This shows that our method could be suitable for previewing solutions of global illumination.

Figure 5.4 demonstrates the progressive rendering of the Conference scene [McGuire 2011]. As can be seen, our method can reduce bright spots near the corners and the curtains in this scene.

The convergence of our method and traditional VPL is shown in Figure 5.5. The plot shows

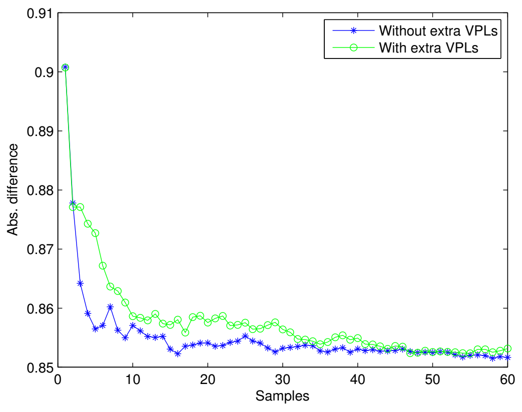


(a) Progressive rendering. First row: ordinary VPLs only. Second row: with extra VPLs.

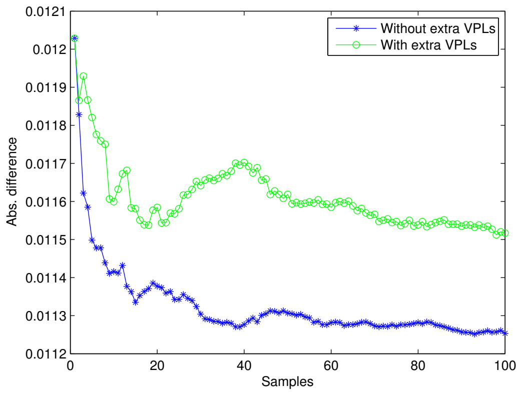


(b) The scene rendered with 400K VPLs.

Figure 5.4: Progressive rendering of the Conference scene [McGuire 2011]. Similarly, our method allows progressive rendering with less bright spots.



(a) The Kitchen scene



(b) The Conference scene

Figure 5.5: The error plot of our tested scenes. The horizontal axis represents the total number of VPLs (in thousands). The vertical axis shows the absolute difference with the ground truth generated by path tracing.

that the bias created by extra VPLs is negligible and our method can converge comparably as the traditional VPL method. In the Kitchen scene, the extra VPLs are about 30% of the total VPLs. In the Conference scene, this ratio is about 50%, which explains the slightly higher bias.

In terms of performance, our method, which is a hybrid CPU and GPU implementation, runs in comparable speed with standard VPL gathering in most of our scenes, and hence is still faster than VSL gathering. The additional clamping map output, pixel clustering, and extra VPL sampling cost a few seconds after each rendering pass. On average, our method is about 1.3x-2x slower than standard VPL, and 3x-5x faster than VSL.

5.5 Conclusions

We proposed an adaptive VPL sampling approach that aims to reduce artifacts in progressive VPL rendering. Our method can be easily integrated with existing VPL gathering framework. It works efficiently and the performance is generally on par with the standard VPL method.

Currently, there are some limitations in this framework. The sampling process can be biased and some regions might be repeatedly examined. While the artifacts can be reduced, they are not completely discarded. Future works include investigating how we can use multiple importance sampling to make the combination of ordinary and extra VPLs more efficient.

Chapter 6

Direct and progressive reconstruction of dual photography images

In this chapter, we start to explore light transport in the real world. In inverse light transport, an important task is to efficiently acquire the light transport of a scene. To achieve this, we use a projector-camera system. When the light transport is acquired, it can be used for dual photography, a well-known application of light transport that can synthesize images from the viewpoint of the projector.

Compressive dual photography [Sen and Darabi 2009] is a fast approach to acquire the light transport for dual photography using compressive sensing. However, the reconstruction step in compressive dual photography can still take several hours before dual images can be synthesized because the entire light transport needs to be reconstructed from measured data. In this chapter, we present a novel reconstruction approach that can directly and progressively synthesize dual images from measured data without the need of first reconstructing the light transport. We show that our approach can produce high-quality dual images in the order of minutes using only a thousand of samples. Our approach is most useful for previewing a few dual images, e.g., during light transport acquisition. As a by-product, our method can also perform low-resolution relighting of dual images. We also hypothesize that our method is applicable to reconstructing dual images in a single projector and multiple cameras system.

6.1 Dual photography

Light transport [Ng et al. 2004] is a mathematical operator that captures how light bounces among surface points in a scene. In computer graphics and computer vision, several applications have been proposed that make use of light transport such as relighting [Ng et al. 2004], dual photography [Sen et al. 2005], and radiometric compensation. Among those, dual photography is an interesting and well-known application of light transport thanks to its simplicity and usefulness. Given a light transport of a scene lit by a controlled light source and captured by a camera, dual photography can virtually swap the roles of the light source and the camera to produce dual images. The dual images can be perceived as if the scene is lit by the camera and captured by the light source. Dual photography is also useful in capturing 6D light transport [Sen et al. 2005].

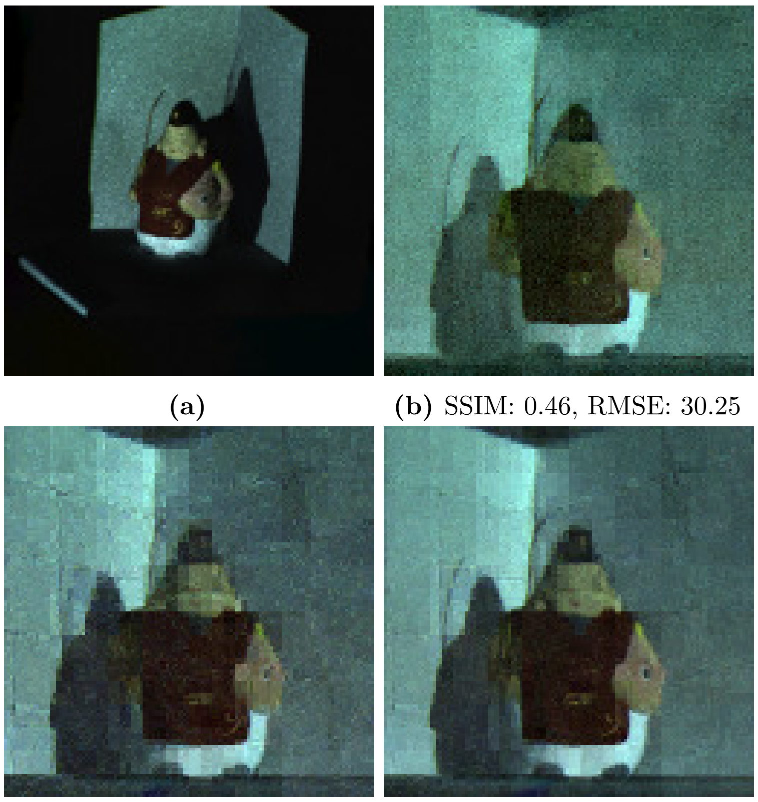
Traditionally, to obtain dual images of a scene, it is necessary to first acquire and reconstruct the entire light transport matrix of the scene. Several approaches have been proposed to efficiently acquire and reconstruct light transport such as multiplexed illumination [Schechner et al. 2003], compressive sensing of light transport [Sen and Darabi 2009; Peers et al. 2009], or optical computing of light transport [O'Toole and Kutulakos 2010]. However, reconstructing the entire light transport matrix from the acquired data can still be very costly since the number of rows and columns of the light transport matrix can be tens or hundreds of thousands. This means a huge amount of computational time is required before the first dual image can be synthesized and ready for display.

In this chapter, we present a novel approach to efficiently compute dual images from measured data without reconstructing the light transport. We build our method upon compressive sensing of light transport [Sen and Darabi 2009; Peers et al. 2009] and propose an approach to directly and progressively reconstruct high-quality dual images using L1-norm optimization. The number of measurement samples needed is comparable to that used for light transport reconstruction in compressive dual photography. Such direct reconstruction allows us to quickly synthesize dual images as soon as the acquisition data is enough. Our method can also generate progressive results while the dual image is being reconstructed. Therefore, our method can be beneficial for previewing a few dual images. Besides, we also demonstrate that our method can be used for low-resolution relighting of dual images. We also hypothesize that our approach is extendable to synthesize dual images in setups that have a single light source and multiple cameras.

6.2 Related works

Recently, several approaches to efficiently acquire and reconstruct light transport have been proposed [Schechner et al. 2003; Sen et al. 2005; Sen and Darabi 2009; Peers et al. 2009; O'Toole and Kutulakos 2010]. In the seminal work about dual photography, Sen et al. [2005] proposed a hierarchical approach to detect projector pixels that can be turned on simultaneously in a single light pattern. This greedy-like approach can reduce the number of light patterns in the acquisition to the order of thousands. In the worst case when most of projector pixels conflict to each other and can only be scheduled to be turned on sequentially, this approach can be as slow as brute-force acquisition.

In compressive dual photography [Peers et al. 2009; Sen and Darabi 2009], the authors proposed to use rows of measurement matrices in compressive sensing as light patterns, thus turns light transport acquisition into a compressive sensing problem that allows the light transport to be reconstructed using the well-known L1-norm optimization. This approach


(a) (b) SSIM: 0.46, RMSE: 30.25

(c) SSIM: 0.47, RMSE: 25.93 (d) SSIM: 0.55, RMSE: 27.32

Figure 6.1: Dual photography. (a) Camera view. (b) Dual image directly reconstructed from 16000 samples, which is not practical. (c) Dual image progressively reconstructed from only 1000 samples using our method with 64 basis dual images. (d) Dual image reconstructed with settings as in (c) but from 1500 samples. Haar wavelet is used for the reconstruction.

works well for high-rank and sparse light transport matrix which is often seen in a projector-camera system. In this work, we also build our approach based upon compressive sensing. We provide a simple reformulation of compressive dual photography that allows us to directly and progressively reconstruct dual images using L1-norm optimization.

Recently, O'Toole and Kutulakos [2010] proposed to use Arnoldi iterations to determine eigenvectors of a light transport using optical computing. While their method only requires less than a hundred of images, it is more suitable for dense and low-rank light transport where the light source is diffuse. In this work, we target sparse and high-rank light transport.

While compressive sensing of light transport is designed to minimize the number of images to acquire, it often results in long computation time needed to reconstruct the light transport in the post-processing step. This is an issue for dual photography, especially when we only need to see a handful number of dual images. Therefore, it is necessary to have an approach that can compute dual images from measured data as fast as possible. In this work, we fill in this gap by proposing such an approach based on compressive sensing and L1-norm optimization.

Sen and Darabi [2009] also discussed about single pixel imaging and how it is related to compressive dual photography. This is probably most closely related to direct reconstruction of dual images which we proposed in this work. The authors noticed that directly recovering the reflectance function of this single pixel, which is equivalent to directly computing the dual image under floodlit lighting, is rather troublesome because the dual image is more complicated and therefore a lot more samples are needed. In this work, we solve this problem by presenting a simple basis so that dual images can be progressively reconstructed from a small amount of measurement samples.

Finally, while it is not closely related to dual photography, we note that the idea of direct reconstruction using compressive sensing was also exploited to obtain the inverse light transport [Chu et al. 2011].

6.3 Compressive dual photography

Let $\mathbf{T}$ be the light transport matrix of a scene captured by a projector-camera system. Suppose that the light source emits pattern $\mathbf{l}$. The image $\mathbf{c}$ of the scene captured by the camera can be represented by the light transport equation:

$$\mathbf{c} = \mathbf{T}\mathbf{l}. \tag{6.1}$$

In dual photography, by utilizing Helmholtz reciprocity, the dual image can be computed as

$$\mathbf{c}' = \mathbf{T}^\top \mathbf{l}', \tag{6.2}$$

where $\mathbf{l}'$ is the dual light pattern virtually emitted by the camera and $\mathbf{c}'$ is the dual image virtually captured by the light source.

By projecting a set of N light patterns $\mathbf{L} = [\mathbf{l}_1 \dots \mathbf{l}_N]$ and capturing images of the scene $\mathbf{C} = [\mathbf{c}_1\ \mathbf{c}_2 \dots \mathbf{c}_N]$ lit by this set of patterns, we can rewrite the light transport equation as

$$\mathbf{C}^\top = \mathbf{L}^\top \mathbf{T}^\top, \tag{6.3}$$

which suggests an elegant way to measure light transport $\mathbf{T}$ using compressive sensing. Each row of $\mathbf{T}$ can be measured by letting $\mathbf{L}^\top$ be a measurement matrix such as Bernoulli or Gaussian matrix that satisfies the restricted isometry property [Baraniuk 2007]. Each row $\mathbf{t}_i$ of light transport matrix $\mathbf{T}$ can be independently reconstructed by minimizing

$$\mathbf{t}_i = \arg\min_{\mathbf{u}} \|\mathbf{c}_i^\top - \mathbf{L}^\top \mathbf{u}\|_2^2 + \lambda \|\mathbf{W}^\top \mathbf{u}\|_1 \tag{6.4}$$

where $\mathbf{c}_i^\top$ denotes column i of $\mathbf{C}^\top$, $i \in [1 \dots |\mathbf{T}|]$, $|\mathbf{T}|$ the number of rows of matrix $\mathbf{T}$, $\mathbf{W}$

the basis of the space where each row of the transport matrix can be sparse. However, since $|\mathbf{T}|$ can be tens of thousands, e.g., $|\mathbf{T}| = 128 \times 128$ which represents a rather low-resolution camera view, the reconstruction of $\mathbf{T}$ can take several hours to complete [Sen and Darabi 2009].

To speed up, it is possible to further exploit coherency among pixels in each column of matrix $\mathbf{T}$ by using another compression basis $\mathbf{P}$ as in [Peers et al. 2009]. We get:

$$\mathbf{P}^\top \mathbf{C} = (\mathbf{P}^\top \mathbf{T} \mathbf{W})(\mathbf{W}^\top \mathbf{L}). \tag{6.5}$$

We capture images as before but transform them into basis $\mathbf{P}$ in the post-processing. As before, compressive sensing can be applied to reconstruct each row of the compressed matrix $\mathbf{P}^\top \mathbf{T} \mathbf{W}$ independently, but this time the number of rows needed to reconstruct can be less. However, in our observation, the number of non-zero rows of $\mathbf{P}^\top \mathbf{C}$ is still in the order of thousands because the captured images $\mathbf{C}$ lit by measurement patterns $\mathbf{L}$ can contain a lot of complex blocky patterns that are difficult to compress by basis $\mathbf{P}$.

6.4 Direct and progressive reconstruction

6.4.1 Direct reconstruction

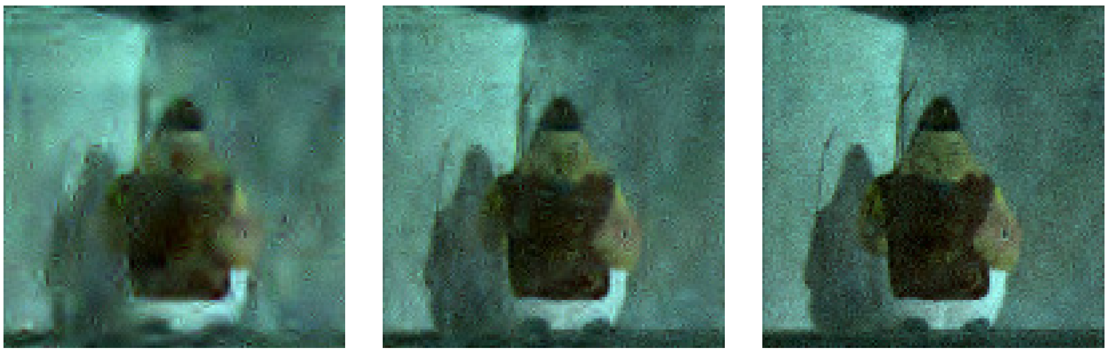
We are now ready to present our approach to directly reconstruct dual images, which we build on top of compressive dual photography [Sen and Darabi 2009]. We start by showing that dual image can be directly computed from the acquired images and light patterns. By multiplying the dual light pattern $\mathbf{l}'$ to both sides of Equation 6.3, it is easy to get:

$$\mathbf{C}^\top \mathbf{l}' = \mathbf{L}^\top \mathbf{c}'. \tag{6.6}$$

By letting $\mathbf{L}^\top$ be a measurement matrix and pre-computing the left part $\mathbf{C}^\top \mathbf{l}'$, we can view dual image synthesis as a compressive sensing problem. Therefore, the dual image can be directly reconstructed by L1-norm optimization:

$$\mathbf{c}' = \arg\min_{\mathbf{u}} \|\mathbf{C}^\top \mathbf{l}' - \mathbf{L}^\top \mathbf{u}\|_2^2 + \lambda \|\mathbf{W}^\top \mathbf{u}\|_1. \tag{6.7}$$

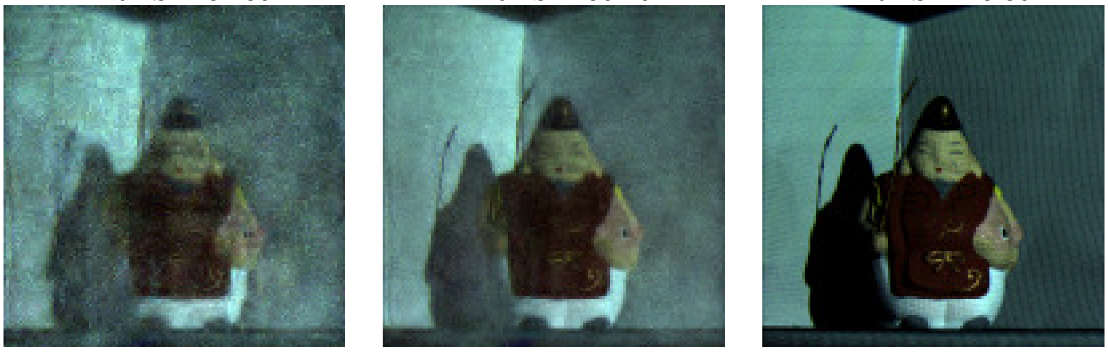
Theoretically, this approach should be able to reconstruct the dual image $\mathbf{c}'$. Unfortunately, in practice, in order to obtain a high-quality dual image, almost tens of thousands number of measurement samples, or camera images and light patterns, are necessary. This is because dual image is not as sparse as reflectance functions stored in rows of light transport $\mathbf{T}$, thus it requires more samples in the reconstruction.



(a) 4000 samples.
SSIM: 0.43
RMSE: 34.35

(b) 8000 samples.
SSIM: 0.44
RMSE: 30.16

(c) 16000 samples.
SSIM: 0.44
RMSE: 28.30



(d) 1000 samples.
SSIM: 0.41
RMSE: 31.86

(e) 2000 samples.
SSIM: 0.57
RMSE: 28.82

(f) Ground truth.

Figure 6.2: Comparison between direct and progressive reconstruction. Dual image (a), (b), and (c) are from direct reconstruction. Dual image (d) and (e) are from progressive reconstruction with 64 basis dual images. (f) Ground truth is generated from light transport from 16000 samples by inverting the circulant measurement matrix. Daubechies-8 wavelet is used for the reconstruction.

6.4.2 Progressive reconstruction

We propose a simple approach in order to overcome the above issue. Suppose that we can project the dual light pattern $\mathbf{l}'$ into a basis $\mathbf{Q} = [\mathbf{q}_1 \; \mathbf{q}_2 \ldots \mathbf{q}_{|\mathbf{Q}|}]$:

$$\mathbf{l}' = \mathbf{Q}\mathbf{w} = \sum_i w_i \mathbf{q}_i, \tag{6.8}$$

where $|\mathbf{Q}|$ is the number of basis vectors in $\mathbf{Q}$, $i \in [1 \ldots |\mathbf{Q}|]$, $\mathbf{w}$ the coefficient vector of $\mathbf{l}'$ in basis $\mathbf{Q}$. Therefore, the dual image can be computed by

$$\mathbf{c}' = \sum_i w_i \mathbf{c}'_i \tag{6.9}$$

where $\mathbf{c}'_i$ is the *basis dual image* which satisfies

$$\mathbf{C}^\top \mathbf{q}_i = \mathbf{L}^\top \mathbf{c}'_i. \tag{6.10}$$

Each basis dual image can be found independently by optimizing

$$\mathbf{c}'_i = \arg\min_{\mathbf{u}} \|\mathbf{C}^\top \mathbf{q}_i - \mathbf{L}^\top \mathbf{u}\|_2^2 + \lambda \|\mathbf{W}^\top \mathbf{u}\|_1. \tag{6.11}$$

The intuition behind this formulation is that we can split the reconstruction of the dual image into several passes, and reconstruct each basis dual image that forms a part of the dual image in each pass. It is significant to guarantee that each basis dual image should be sufficiently sparse so that it can be successfully reconstructed using Equation 6.11 without using too many measurement samples. As shown in Figure 6.1 and 6.2, the number of samples needed to reconstruct basis dual images is comparable to that required to reconstruct the entire light transport in traditional compressive dual photography, which is more practical than direct reconstruction. Figure 6.3 shows a few examples of the progressive reconstruction.

We choose basis $\mathbf{Q}$ based on two following criteria. First, the dimension of space $\mathbf{Q}$ should be as low as possible. It is best to choose $\mathbf{Q}$ of which the dimension is about tens or hundreds. Second, the basis dual images $\mathbf{c}'_i$ obtained by setting dual lighting to basis vectors of $\mathbf{Q}$ should be sparse so that high quality reconstruction can be achieved.

Based on such criteria, we propose a simple and easy to implement basis $\mathbf{Q}$ as follows. We subdivide the dual lighting pattern $\mathbf{l}'$ into a grid and let each patch in the grid be a basis vector $\mathbf{q}_i$. Therefore, the weight w_i is simply set to one. It is easy to see that smaller patch size tends to produce sparser coefficients of basis dual images in the wavelet domain. This can yield higher accuracy in the reconstruction but result in longer computational time.

An advantage of choosing basis $\mathbf{Q}$ as above is that we can display progressive results of the dual image by accumulating existing basis dual images while other remaining basis dual images are pending for reconstruction, which is useful for previewing applications.

6.5 Implementation

We use a projector-camera system to acquire the light transport. The projector is a Sony VPL-DX11. The camera is a Sony DXC-9000 of which the response curve is linear.

The light patterns to compressively acquire the light transport are obtained from a circulant matrix of which the first row is an i.i.d Bernoulli distribution with value -1 and 1 [Yin et al. 2010]. An advantage of using a circulant measurement matrix is that its multiplication with a vector can be quickly computed using fast Fourier transform. Also, circulant matrix requires very little memory storage as only the first row needs to stored.

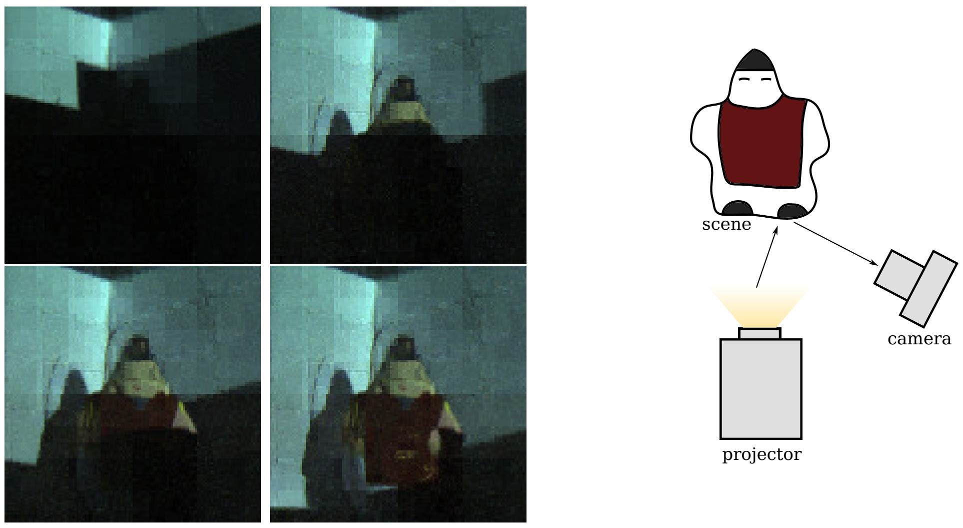


Figure 6.3: Progressive results of the dual image in Figure 6.1(d) by accumulating those reconstructed basis dual images. Our projector-camera setup to acquire light transport is shown in the diagram.

Since our patterns contain both positive and negative values, we project positive and negative patterns separately and combine the corresponding camera images in the post-processing by the formula $\mathbf{c} = \mathbf{T}(\mathbf{l}^{+} - \mathbf{l}^{-}) = \mathbf{c}^{+} - \mathbf{c}^{-}$, where superscript $+$ and $-$ denote positive and negative patterns and images, respectively. For simplicity, we also crop and downsample camera images to the same size as the light patterns so the light transport is a square matrix.

We implement our system in MATLAB. We implement split Bregman iterations [Goldstein and Osher 2009] for L1-norm optimization in Equation 6.11. We let $\lambda = 0.001$ for all progressive reconstruction. We let $\lambda = 0.05$ for direct reconstruction to further suppress noise. We test the reconstruction with Haar wavelet and Daubechies-8 wavelet provided by the Rice Wavelet Toolbox [Baraniuk 2002].

During progressive reconstruction, we discard basis dual images of which the absolute maximum value of their corresponding left-hand side vector $\mathbf{C}^{\top}\mathbf{q}_i$ is less than 10^{-4}. In fact, this corresponds to regions that can be lit by the projector but are out of field of view of the camera so zero solutions for basis dual images are appropriate.

6.6 Experiments

The results of our method are shown in Figure 6.1. The resolution of the dual image is 128×128. As can be seen, our method is able to reconstruct a good-quality dual image without first obtaining the light transport. We provide quantitative comparisons between our results of direct and progressive reconstruction and the ground truth shown in Figure 6.2

using both structural similarity index (SSIM) [Wang et al. 2004] and root-mean-square error (RMSE).

In Figure 6.1, by using basis $\mathbf{Q}$ with patch size set to 16 pixels, only 1000 samples are needed to reconstruct total 64 basis dual images and a high-quality final dual image. Figure 6.3 shows some of the progressive results of the dual image during reconstruction. In contrast, directly reconstructing the dual image without basis $\mathbf{Q}$ requires 16000 samples in order to reach similar image quality, which is far less practical. In fact, given 16000 samples, it is often more preferable to reconstruct the entire light transport in the post-processing by inverting the circulant measurement matrix using FFT, which is fast. Here we use this approach to generate the ground truth dual image as shown in Figure 6.2(f). We do not opt to reconstruct the light transport from only 1000 measurement samples since it takes tens of thousands of L1-norm optimization that is too time consuming to perform.

Figure 6.2 further demonstrates how our method works with different number of samples for both direct and progressive reconstruction using Daubechies-8 wavelet for compression. As expected, more samples allows more details of the dual image to be revealed.

As a by-product, we demonstrate a relighting application by linearly combining basis dual images by setting the weight vector to a low-resolution lighting pattern. Figure 6.4 shows our relit images. The new lighting has resolution 8×8 since our basis vectors are derived from 8×8 grid patches.

We measured the running time of our progressive reconstruction on an Intel Core 2 Quad processor clocked at 2.8 GHz with 8 GB of RAM. Our MATLAB implementation output the direct result (b) of Figure 6.1 in 10 minutes and the progressive result (c) in 40 minutes. While progressive reconstruction is a few times slower, it saves a large amount of acquisition time as it requires far less number of samples to reach similar image quality. With the same number of samples, progressive reconstruction is also faster than reconstructing the entire light transport when only a few images are needed.

6.6.1 Running time analysis

We provide a simple analysis to estimate how much and when progressive reconstruction is better than traditional light transport reconstruction in terms of running time as follows. We assume the following model to predict the running time of progressive reconstruction:

$$t = 2\alpha N + k\rho|\mathbf{Q}|, \tag{6.12}$$

where t is the running time in seconds, N the number of samples acquired, α the time to acquire a single image, ρ the time to reconstruct a basis dual image, k the number of dual images we are interested in in total. The constant 2 represents the need to capture two images per sample due to positive and negative entries of the measurement matrix. Similarly,

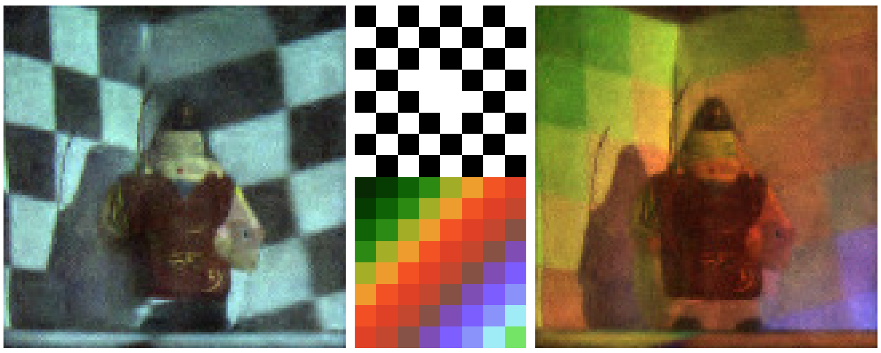


Figure 6.4: Relighting of the dual image in Figure 6.2(e).

the running time of traditional light transport reconstruction can be predicted by:

$$t' = 2\alpha N' + \rho' |\mathbf{T}|, \tag{6.13}$$

where N' is the number of samples needed to acquire for light transport reconstruction, ρ' the time to reconstruct a row of $\mathbf{T}$.

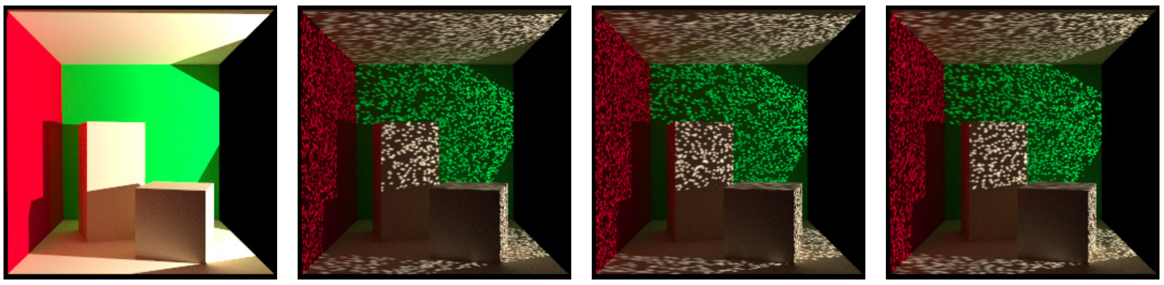
Empirically, we set $\alpha = 1$ second, $N = 1000$ samples, $\rho = 75$ seconds, according to the examples in the previous figures. Since the reflectance function stored in each row of light transport $\mathbf{T}$ can be more sparse than dual images, we pessimistically assume that $N' = 500$ which means our progressive reconstruction requires twice the number of samples. We also set $\rho' = 2$ to assume that each row of $\mathbf{T}$ can be reconstructed much faster. We also have $|\mathbf{Q}| = 64$ and $|\mathbf{T}| = 16000$.

As a result, in order to guarantee $t < t'$, we need to bound $k \leq 6$. This indicates the maximum number of dual images we can reconstruct before our method cannot offer any time savings. When only a dual image is needed, or $k = 1$, the speed up is about $5\times$.

6.7 More results

In this section, we present another progressive dual image reconstruction example from a synthetic light transport of a Cornell box scene. The light transport is generated in LuxRender using path tracing with approximately 1024 samples per pixel. It is easy to see that the dual image reconstructed is correct and consistent with the original camera view.

In this example, we set the patch size to 16 to increase sharpness of the dual image. As the image size is 256×256, there are in total 256 basis images. It takes roughly two hours for the progressive reconstruction to complete. While this example needs longer time to reconstruct the dual image than the previous example, this is still much quicker than reconstructing the whole light transport, which may need to reconstruct up to 64K reflectance functions.


(a)

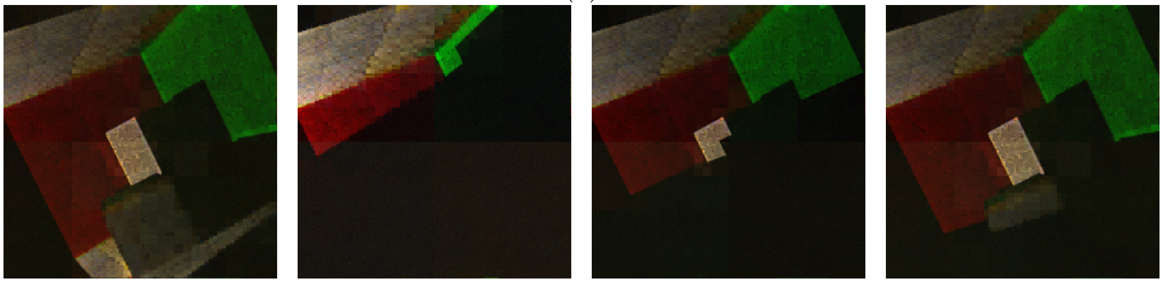

(b)

Figure 6.5: Dual photography. (a) Camera view and generated images for capturing light transport. The projector is on the right of the box. (b) Dual image and the progressive reconstruction (floodlit lighting) from 4000 samples using our method with 256 basis dual images. Haar wavelet is used for the reconstruction. Image size is 256×256.

6.8 Discussion

Conventionally, in order to compute a dual image of light transports of a scene captured by a single projector and multiple cameras, the light transport matrix between each pair of projector-camera needs to be reconstructed. In such case, for quick reconstruction, our method is still applicable. In the case of two cameras, we have:

$$\begin{bmatrix} \mathbf{C}_1^\top & \mathbf{C}_2^\top \end{bmatrix} \begin{bmatrix} \mathbf{l}'_1 \\ \mathbf{l}'_2 \end{bmatrix} = \mathbf{L}^\top \mathbf{c}'. \tag{6.14}$$

It is natural to extend the formulation to the case of multiple cameras. We leave the implementation of such a system for future works.

6.9 Conclusions

In this chapter, we presented an approach based on compressive sensing to directly and progressively reconstruct dual photography images without the need of reconstructing the entire light transport. Our method can be useful for previewing of dual images. We are also able to perform low-resolution relighting of dual images.

There are a few limitations in our approach. First, our reconstructed dual images tend to be noisier than those produced by the full light transport. This can be explained by the dot product between the camera images and the dual lighting pattern, which sums up the variance of each camera pixel. Second, our method may fail when the basis dual images are not sparse enough.

It is interesting to extend this work further in the future. First, it can be useful to have a careful noise analysis of dual images obtained by our method. Second, it can be exciting to seek a more optimal basis than our grid basis in order to reconstruct dual images in higher quality.

Chapter 7

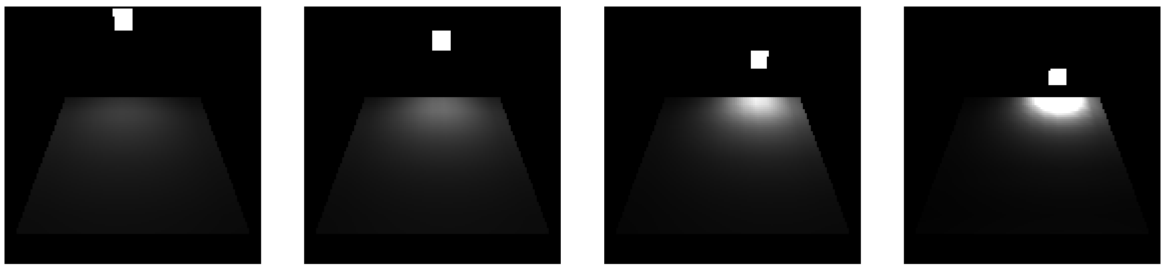
Reconstruction of depth and normals from interreflections

From the previous chapter, we see that the light transport matrix of a real scene can be efficiently acquired using compressive sensing. Subsequently, it is therefore desirable to extract scene information from this matrix, e.g., surface geometry and materials. In this chapter, we explore how to reconstruct geometry from a light transport.

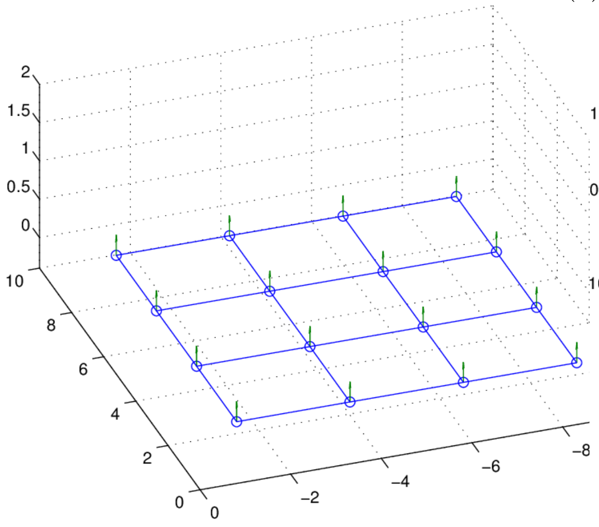
While geometry reconstruction has been extensively studied, several shortcomings still exist. First, traditional geometry reconstruction methods such as geometric or photometric stereo only recover either surface depth or normals. Second, such methods require calibration. Third, such methods cannot recover accurate geometry in the presence of interreflections. In order to address these problems in a single system, we propose an approach to reconstruct geometry from light transport data. Specifically, we investigate the problem of geometry reconstruction from interreflections in a light transport matrix. We show that by solving a system of polynomial equations derived directly from the interreflection matrix, both surface depth and normals can be fully reconstructed. Our system does not require projector-camera calibration, but only make use of a calibration object such as a checkerboard in the scene to pre-determine a few known points to simplify the polynomial solver. Our experimental results show that our system is able to reconstruct accurate geometry from interreflections up to a certain noise level. Our system is easy to set up in practice.

7.1 Geometry from light transport

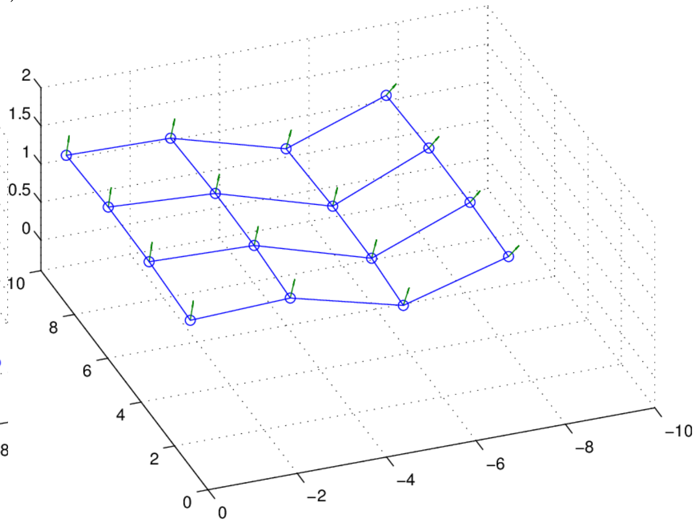
Geometry reconstruction has been extensively studied in computer vision in the past decades. Reconstruction techniques such as geometric stereo and photometric stereo have greatly matured, and have widely been used in both scientific and industrial applications. However, like many other computer vision techniques, previous reconstruction approaches only account for direct illumination and ignores an important lighting effect that often occurs in a scene: global illumination. Therefore, those techniques can only handle scenes in which interreflection or sub-surface scattering is absent. In order to improve robustness of geometry



(a)



(b)



(c)

Figure 7.1: (a) Synthetic light transport using radiosity. (b) Reconstructed points from exact data by form factor formula. (c) Reconstructed points from data by radiosity renderer.

reconstruction, global illumination would need to be properly considered.

4D light transport is a general matrix representation that captures a scene observed in a set of varying illuminations. An entry in the matrix captures the out-going radiance at a scene point illuminated by a light source. It is also well-known that under Lambertian assumption, light transport matrix can be factorized into the first-bounce light transport matrix which captures direct illumination, and the interreflection matrix which captures illumination that bounces from a surface to another in the scene [Seitz et al. 2005]. In computer graphics, several applications of light transport have been proposed such as relighting, dual photography, and radiometric compensation. However, in computer vision, light transport has not been received great attentions for tasks such as geometry reconstruction. Since light transport captures global illumination, it is of great interest to explore geometry reconstruction from such global illumination data.

In this work, we present a new approach to recover scene geometry from light transport. Our reconstruction is based on solving a system of polynomial equations derived directly from the interreflection matrix. We show that our method can reconstruct both surface depth and normals from interreflections. Our method does not require the projector and the camera to be calibrated. It also does not rely on orthographic assumption and planar constraints [Liu et al. 2010]. We only use a checkerboard pattern in the scene to pre-determine coordinates of a few points to bootstrap the solving of polynomial equations. Therefore, it can more easy to use in practice.

7.2 Related works

In this section, we first discuss two classes of traditional reconstruction techniques, triangulation-based methods and photometric stereo methods. We then discuss about recent techniques that recover geometry in the presence of global illumination.

7.2.1 Conventional methods

Triangulation-based methods, e.g., geometric stereo and structured light scanning, has long been common approaches for geometry reconstruction. Geometric stereo is sometimes problematic since it relies on scene features such as corners to determine correspondences, which is not always robust. Structured light scanning projects special light patterns into the scene so that correspondences between the projector and the camera can be decoded in the post-process. However, while triangulation-based methods yields 3D coordinates of scene points, it does not compute surface normals directly. Surface normals can be found from derivatives of local surfaces that needs to be reconstructed for each neighborhood of scene points.

On the other hand, photometric stereo observes the scene under varying illumination with the camera view fixed. Based on surfaces illuminated by at least three different directional light sources, surface normals can be solved from a linear system. In contrast to triangulation-based methods, photometric stereo yields surface normals directly, but it does not compute 3D coordinates of surface points. 3D coordinates can be determined by integrating normal vectors. Since triangulation-based methods and photometric stereo reconstruct attributes of surfaces that are complementary to each other, it is of great interest to seek methods that can produce surface depth and normals at the same time. In this work, we propose such an approach that aims to reconstruct geometry from light transport.

In addition, a common drawback of conventional geometric and photometric stereo is that calibration is necessary. Geometric stereo requires the camera to be calibrated while photometric stereo assumes directional light source and requires the directions of the light sources to be known. Some efforts has been done to relax the necessity of such calibration. For example, Basri et al. [2007] showed that surface normals can be recovered from uncalibrated photometric stereo up to a general bas-relief ambiguity. Recently, Yamazaki et al. [2011]

proposed the joint recovery of intrinsic and extrinsic parameters of both camera and projectors in a projector-camera setup. However, their method still requires the center of projection of both camera and projector to be known.

Extensions of photometric stereo to near point light source have also been proposed [Iwahori et al. 1990; Kim and Burger 1991]. In such setup, depth recovery can be incorporated into photometric stereo due to the modeling of light fall-off by the inverse squared law. However, while near point light source is more practical, these methods still require the location of the light sources to be known. Our system is more convenient as it does not require the calibration of the projector. The only object that we need is a checkerboard pattern put in the scene to help determine known points in the post-processing.

7.2.2 Hybrid methods

In this work, our proposed system jointly reconstructs surface depth and normals and hence can be regarded as a combination of geometric and photometric stereo in terms of output. In this aspect, several similar hybrid systems have been proposed in the past. For example, Aliaga and Xu [2008] proposed a self-calibration method that utilizes both geometric and photometric stereo. Holroyd et al. [2010] combined multiple view reconstruction and phase shifting to recover complete 3D geometry and surface reflectance of a target object. Yoon et al. [2010] suggested a non-linear optimization framework to recover geometry and reflectance from multiple view geometry, which requires a good initialization for the non-linear optimization. While our system is quite similar to these works, we explore geometry reconstruction from light transport data of a scene. This can be more convenient since light transport can also be at the same time utilized for other applications relighting and radiometric compensation. Our system also does not require explicit calibration as in [Holroyd et al. 2010].

7.2.3 Reconstruction in the presence of global illumination

While traditional reconstruction methods work well for Lambertian and mostly diffuse surfaces, they ignore an important effect that is commonly seen: global illumination. This strict assumption can limit accurate shape reconstruction when global illumination is strong, e.g., when light bounces within concave surfaces. It has been shown that photometric stereo tends to produce a shallower concave surface if interreflections are not taken into account [Nayar et al. 1991].

In order to accurately reconstruct geometry in the presence of global illumination, two different strategies can be used. The first approach is to separate global illumination based on the principle proposed by Nayar et al. [2006]. They show that since global illumination is a low-frequency effect, it is almost invariant to high-frequency illumination. Therefore, by using high-frequency light patterns, either binary or phase-shift patterns, it is possible to separate direct and global illumination. Since then, several methods have been proposed to make geometry reconstruction robust to global illumination. Gupta et al. [2012] studied the relationship between projector defocus and global illumination and showed that such adverse effects can be separated and removed from the scene. Geometry can then be reconstructed from direct illumination. Gupta et al. [2013] proposed a method to design structured light patterns that yield accurate correspondences in the presence of short-range and long-range global illumination. Gupta and Nayar [2012] also suggested that phase shifting can be extended to include only high-frequency patterns so that reconstruction is robust to global illumination. Couture et al. [2011] showed that random patterns could also be used to finding robust correspondences. However, methods based on explicitly removing global illumination and reconstructing geometry from residual direct illumination can still fail when signal-to-noise ratio of direct illumination is too low, e.g, as in translucent objects that have strong sub-surface scattering. Approaches that do not require explicit removal of global illumination do not have this drawback, but they need different pattern designs to handle different global illumination effects [Gupta et al. 2013]. Furthermore, all these approaches are based on triangulation, which requires the light source and the camera to be fully calibrated.

Another approach to handle global illumination is to model it explicitly, which is also the approach we chose to follow. This class of methods can be useful when the scene is dominated by global illumination. Nayar et al. [1991] proposed to refine surface normals obtained by photometric stereo using interreflection. Liu et al. [2010] proposed to reconstruct geometry from the interreflection matrix. We note that the work in [Liu et al. 2010] is probably most related to ours. However, the authors assumed orthographic projection and did not properly handle the area term in the interreflection model. We show that our method is independent of the type of camera projection, and it can handle the area term properly by considering it as an unknown scalar in the system of polynomials.

In summary, we highlight three shortcomings from previous approaches. First, triangulation-based methods only recover surface depth while photometric stereo only recovers surface normals. Second, traditional geometric and photometric stereo require the acquisition system to be carefully calibrated. Hybrid methods are needed to jointly recover both surface depth and normals. Third, and more importantly, global illumination is often ignored, which can cause reconstruction surfaces to be shallower, as shown in [Nayar et al. 1991]. As far as we know, there has been no single acquisition system that address such shortcomings altogether.

Therefore, in this work, we propose to build an acquisition system that is aimed to fill this gap. Our hybrid system can jointly recovers surface depth and normals. We explore how to reconstruct such depth and normals directly from interreflections in a light transport. Our system does not require orthographic assumption and planar constraints as in [Liu et al. 2010] and does not need calibration. We only use a checkerboard in the scene to determine a few known points in order to simplify the polynomial solver in the reconstruction. Therefore, our system is easier to implement and more convenient to use in practice.

7.3 Interreflections in light transport

The rendering equation that computes the out-going radiance L at scene point $\mathbf{x}$ to scene point $\mathbf{x}''$ can be written as

$$L(\mathbf{x}, \mathbf{x}'') = L_d(\mathbf{x}, \mathbf{x}'') + \int_{\mathbf{x}'} \mathbf{A}(\mathbf{x}', \mathbf{x}, \mathbf{x}'') L(\mathbf{x}', \mathbf{x}) d\mathbf{x}' \tag{7.1}$$

where $\mathbf{A}$ is the interreflection operator, L_d is the direct illumination from $\mathbf{x}$ to $\mathbf{x}''$. We define light transport operator $\mathbf{T}$ that captures the net effect of the whole light transport in the scene as follows.

$$L(\mathbf{x}, \mathbf{x}'') = \int_{\mathbf{x}'} \mathbf{T}(\mathbf{x}', \mathbf{x}, \mathbf{x}'') L_e(\mathbf{x}', \mathbf{x}) d\mathbf{x}', \tag{7.2}$$

where L_e is the emitted radiance from light sources. Similarly, we define the first-bounce light transport $\mathbf{F}$ which only stores direct illumination as

$$L_d(\mathbf{x}, \mathbf{x}'') = \int_{\mathbf{x}'} \mathbf{F}(\mathbf{x}', \mathbf{x}, \mathbf{x}'') L_e(\mathbf{x}', \mathbf{x}) d\mathbf{x}'. \tag{7.3}$$

As we assume Lambertian surfaces, the rendering equation becomes the radiosity equation. Since the out-going radiance is the same for all directions determined by $\mathbf{x}''$, we drop the outgoing direction $\mathbf{x}''$ and simply store radiosity $\pi L(\mathbf{x}, \mathbf{x}')$ at each surface point $\mathbf{x}$. Numerically, a light transport matrix $\mathbf{T}$ can be represented by

$$\mathbf{T} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{F} \tag{7.4}$$

where $\mathbf{I}$ is the identity matrix. Since all surfaces are Lambertian, first-bounce $\mathbf{F}$ and inverse light transport $\mathbf{T}^{-1}$ can be computed from light transport $\mathbf{T}$ as in [Seitz et al. 2005]. The interreflection matrix $\mathbf{A}$ can be obtained by

$$\mathbf{A} = \mathbf{I} - \mathbf{F}\mathbf{T}^{-1}. \tag{7.5}$$

Since the interreflection matrix $\mathbf{A}$ captures how much illumination bounces from a surface to another in the scene, it is possible to utilize such information for geometry reconstruction. We show how it can be done in the following section.

7.4 Geometry reconstruction from interreflections

7.4.1 Polynomial equations from interreflections

Each element $\mathbf{A}_{i\to j}$ (represented as matrix entry $\mathbf{A}_{ji}$) captures how radiosity from a source surface patch i contributes to a target patch j and can be written as:

$$\mathbf{A}_{i\to j} = \mathbf{k}_j \mathbf{G}_{i\leftrightarrow j} \Delta_i \tag{7.6}$$

where $\mathbf{k}_j$ is the albedo of patch j, Δ_i the area of patch i, and $\mathbf{G}_{i\leftrightarrow j} = \mathbf{G}_{j\leftrightarrow i}$ the geometric factor between patch i and patch j:

$$\mathbf{G}_{i\leftrightarrow j} = \frac{\mathbf{n}_i^\top(\mathbf{x}_i - \mathbf{x}_j)\mathbf{n}_j^\top(\mathbf{x}_i - \mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|^4} \tag{7.7}$$

where $\mathbf{x}$ and $\mathbf{n}$ denote the location and orientation of a patch, respectively. If patch i is visible in the camera view, its area can be approximated as:

$$\Delta_i = \Delta_{\text{pixel}} \frac{\|\mathbf{c} - \mathbf{x}_i\|}{\mathbf{n}_i^\top(\mathbf{c} - \mathbf{x}_i)} \tag{7.8}$$

where $\mathbf{c}$ is the camera location and Δ_{pixel} is the area of the pixel that contains patch i.

It is easy to see that the interreflection matrix $\mathbf{A}$ captures albedo, location, and orientation of geometric points in the scene. Our goal is to reconstruct the location and orientation of the geometry from $\mathbf{A}$. However, solving the complete geometry from $\mathbf{A}$ can be very challenging because interreflection equations are non-linear and there are a large number of unknowns. To make the problem tractable, we assume a set of known points Q in the scene and try to reconstruct the set of unknown points P from the interreflections between P and Q.

Consider a pair of points $\mathbf{p}_i$ and $\mathbf{q}_j$ where $i \in P$ and $j \in Q$. We would like to reconstruct the albedo, location, and orientation of $\mathbf{p}_i$ from its interreflection with $\mathbf{q}_j$, which are captured by entries $\mathbf{A}_{i\to j}$ and $\mathbf{A}_{j\to i}$ in the interreflection matrix.

Consider $\mathbf{A}_{i\to j}$. We observe that equation $\mathbf{A}_{i\to j}$ is almost a polynomial except the area term Δ_i that depends on the foreshortening of the patch to the camera view. We now show how to formulate $\mathbf{A}_{i\to j}$ into a polynomial.

For simplicity, we first drop index i since we are going to fix i and only consider $\mathbf{A}_{i\to j}$ for varying j. Therefore, we rewrite Equation 7.6 as

$$\mathbf{A}_j = \mathbf{k}_j \mathbf{G}_{i\leftrightarrow j} \Delta \tag{7.9}$$

Let $\mathbf{a}_j = \mathbf{k}_j \Delta$. We further assume that $\mathbf{k}_j$ is invariant for points $j \in Q$ where $\mathbf{A}_j > 0$. This is a reasonable assumption since we can group points that have similar albedos together into group Q. This allows us to model $\mathbf{a}_j$ as a single scalar variable $a = \mathbf{a}_j$ for all $j \in Q$. Multiplying a with the orientation $\mathbf{n}$ to obtain $\mathbf{m} = a\mathbf{n}$, the radiosity from $\mathbf{q}_j$ to $\mathbf{p}_i$ can be written as:

$$\mathbf{A}_j = \frac{\mathbf{m}^\top(\mathbf{x}-\mathbf{x}_j)\mathbf{n}_j^\top(\mathbf{x}-\mathbf{x}_j)}{\|\mathbf{x}-\mathbf{x}_j\|^4} \tag{7.10}$$

which is a polynomial equation in which the unknowns are a 6-DOF vector $(\mathbf{x}, \mathbf{m})$. We now propose an algorithm to solve $(\mathbf{x}, \mathbf{m})$.

7.4.2 Algorithm to recover location and orientation

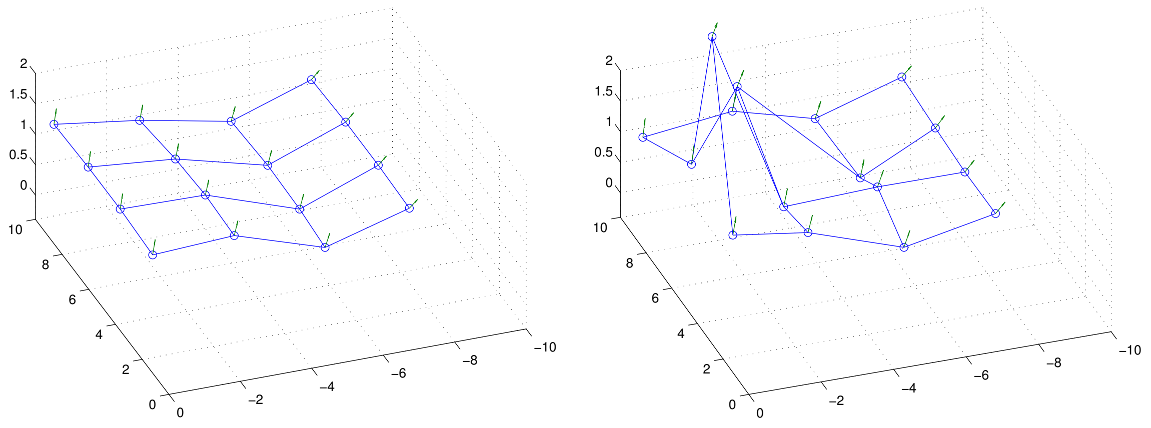
Equation 7.10 suggests that at least six points in Q are necessary to recover each point $\mathbf{p}_i$ separately. The equations can be easily built given the entries $\mathbf{A}_{i\to j}$ for $j \in Q$. Notice that we do not make use of $\mathbf{A}_{i\to j}$ by fixing j and varying i in group P because it is often less practical to assume that the area term Δ_i is constant for different i.

We build an algebraic polynomial solver based on Groebner basis to solve $(\mathbf{x}, \mathbf{m})$. We observe that the solutions given by the algebraic solver are very close to the ground truth, and can be further refined by a non-linear iterative solver when necessary. In general, the algorithm to reconstruct $(\mathbf{x}, \mathbf{m})$ at each point $\mathbf{p}_i$ is as follows.

1. Randomly select six points $\mathbf{q}_j$ s.t. $j \in Q$ and $\mathbf{A}_{i\to j} > 0$.
2. Reconstruct $(\mathbf{x}, \mathbf{m})$ using an algebraic polynomial solver.
3. Compute the residuals from the polynomial equations and repeat the above steps N times. Take $(\mathbf{x}, \mathbf{m})$ that has the lowest residual.
4. Refine $(\mathbf{x}, \mathbf{m})$ with all points $\mathbf{q}_j$ in Q by a non-linear iterative solver.

7.4.3 Implementation

In practice, we implement the above framework with the following assumptions. In Step 1, we assume points in set Q to be planar. Locations and orientations of points on a plane can be easily determined by a simple camera calibration. In Step 2, we translate known points to plane $z = 0$ and orient the plane towards positive Z-axis. We note that this simplifies the Groebner basis of the system of polynomials to a set of 36 monomials. Positioning the plane at other locations can make the system of polynomials more challenging to solve. For example, letting the plane be at $z = \alpha$ that $\alpha \neq 0$ results in a Groebner basis that has 106 monomials. We implement a floating-point polynomial solver based on the action matrix approach. Since there may have several solutions, those that violates visibility constraints are discarded in advance before proceeding to compute residuals. Step 3 is very similar to RANSAC [Fischler and Bolles 1981]. However, here only a few iterations of the first two



(a) Variance 10^{-2} (b) Variance 10^{-1}

Figure 7.2: Reconstruction results with noise variance 10^{-2} and 10^{-1} added to input images.

steps are needed since the result can be refined in Step 4. We use Levenberg-Marquardt optimization [Moré 1978] in Step 4.

7.5 Experiments

We test our algorithm with a synthetic scene rendered by direct form factor calculation and a progressive radiosity algorithm. We use 16 area light sources to individually illuminate a known plane Q. The light sources are distributed uniformly on an unknown plane P and our goal is to reconstruct the locations and orientations of the light sources. For simplicity we only render direct illumination and set albedos of scene objects to one. Therefore, the radiance observed at plane Q can be directly used to find the locations and orientations of light sources on P.

Figure 7.1 demonstrates that our algorithm can successfully reconstruct the locations and orientations of each light sources. We note that our synthetic example is sufficient to test our reconstruction from the system of polynomials. While our algorithm can work with both data from exact form factor and data generated by a radiosity renderer in this example, we did notice a slight shift in the geometry reconstructed from the later as compared to the groundtruth. This can be due to inaccuracy of the intensity values generated by radiosity methods.

In practice, the captured images can be subject to noise. In order to test how our method behaves to noise in this synthetic scenario, we proceed to add Gaussian noise to observed pixel values. Figure 7.2 shows that our solver can tolerate a certain amount of noise with variance up to 10^{-1}.

We acknowledge that since our method relies on radiometric values, i.e., radiance, and numerical solvers for reconstruction, our recovered geometry can be susceptible to noise and may not be as accurate as traditional methods that bases on triangulation.

7.6 Conclusions

We proposed a novel approach to acquire geometry from interreflections. A system of polynomial equations is established directly from the interreflection matrix and we show that by solving this system of polynomial equations, the geometry of the scene, i.e., surface depth and normal vectors, can be jointly reconstructed. Our experimental results demonstrated that our method works well with synthetic datasets up to a certain noise level. Our system is convenient since it does not require calibration.

Our system is limited by the following factors. First, while projector and camera calibration are not needed, a planar checkerboard must be placed in the scene and interact with scene objects in order to simplify the polynomial system. This can cause the arrangement of objects in the scene to be not flexible. Second, our system can be susceptible to noise. The floating-point implementation of the solver of polynomial equations may return wrong solutions when the input data is perturbed by a small amount of noise. Third, our model is based on Lambertian assumption. In practice, this assumption may not be always true. Surfaces in the scene can be up to some certain degrees of glossiness, which violates the interreflection model and causes the system to fail to reconstruct the geometry. Finally, since we rely on acquiring light transport and solving polynomials for geometry reconstruction, our system is not fast enough for real-time reconstruction.

From this study, we recognize several open problems for future research. A potential direction is to design reconstruction methods for more general materials, e.g., glossy or sub-surface scattering surfaces. It is more challenging to fully model such effects than to model diffuse interreflections. Moreover, extracting the global illumination matrix in such cases can be more difficult if the first-bounce matrix is not given. One of the first works in this direction, e.g., shape from translucent surfaces, has been proposed in [Inoshita et al. 2012]. Another potential direction can be to investigate the stability of the polynomial solver used in our approach. In this work, we only used the simplest form of the floating-point implementation of a polynomial solver. We hypothesize that the solver can perform better if stability approaches can be added [Byrod et al. 2009]. Finally, it is of great interest to study fast light transport acquisition to accelerate the data capturing stage and make the system more practical. We also would like to perform more physical experiments to further test our whole proposed pipeline thoroughly, since in this work we only present synthetic examples.

Chapter 8

Conclusions

This thesis explores forward and inverse light transport. In the first part, many-light rendering is studied. Two important problems in many-light rendering, importance sampling using virtual lights, and artifact removal are investigated. Our experiments demonstrated that our proposed solutions are effective. In the second part, two problems in light transport acquisition and analysis are addressed and solutions to these problems were implemented successfully.

While this thesis studies both forward and inverse light transport, bridging the gap between these two areas would definitely need further research. While both of the areas have light transport and the light transport matrix to be the common factor, problems in each area requires different fundamental techniques to address. For example, in forward rendering, one generally uses Monte Carlo integration, the rendering equation, and rendering algorithms such as path tracing, photon mapping, and many-light rendering. In inverse light transport, one needs hierarchical clustering, compressive sensing, and optimization techniques. It is therefore quite challenging to bring such seemingly separate and independent problems into a unified framework. Forward rendering seldom directly uses the raw form of light transport acquired from real world, and inverse light transport requires more technical advances to build scenes and render high-quality images from real-world light transport matrix efficiently.

This thesis leads to a few important open problems to explore. First, in forward light transport, many-light rendering can be integrated into existing Monte Carlo path tracing algorithms and guide the algorithms to converge faster. Adapting many-light rendering techniques to real-time applications is also challenging. Second, in inverse light transport, indirect illumination is a good source of information for geometry and material. It would be interesting to investigate material acquisition from indirect illumination. Finally, it is interesting to ask the question if there exists a sampling approach that can both be used to construct the light transport in both forward and inverse rendering.

References

Aila, T. and Laine, S. 2009. Understanding the efficiency of ray traversal on gpus. In *Proceedings of the Conference on High Performance Graphics 2009*. HPG '09.

Aliaga, D. G. and Xu, Y. 2008. Photogeometric structured light: A self-calibrating and multi-viewpoint framework for accurate 3d modeling. In *Computer Vision and Pattern Recognition (CVPR)*.

Baraniuk, R. 2002. Rice wavelet toolbox.

Baraniuk, R. 2007. Compressive sensing. *IEEE Signal Processing Mag*, 118–120.

Basri, R., Jacobs, D., and Kemelmacher, I. 2007. Photometric stereo with general, unknown lighting. *International Journal of Computer Vision (IJCV) 72*, 239–257.

Birn, J. 2014. Lighting challenges.

Burke, D., Ghosh, A., and Heidrich, W. 2005. Bidirectional importance sampling for direct illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*. EGSR'05.

Byrod, M., Josephson, K., and Astrom, K. 2009. Fast and stable polynomial equation solving and its application to computer vision. *International Journal of Computer Vision (IJCV) 84*, 237–256.

Chu, X., Ng, T.-T., Pahwa, R., Quek, T. Q., and Huang, T. 2011. Compressive inverse light transport. *Proceedings of the British Machine Vision Conference*.

Cohen, M. F., Chen, S. E., Wallace, J. R., and Greenberg, D. P. 1988. A progressive refinement approach to fast radiosity image generation. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '88.

Couture, V., Martin, N., and Roy, S. 2011. Unstructured light scanning to overcome interreflections. In *International Conference on Computer Vision (ICCV)*. 1895 –1902.

Dachsbacher, C., Křivánek, J., Hasan, M., Arbree, A., Walter, B., and Novak, J. 2014. Scalable realistic rendering with many-light methods. *Computer Graphics Forum 33,* 1, 88–104.

Dachsbacher, C. and Stamminger, M. 2005. Reflective shadow maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*. I3D '05.

Dachsbacher, C. and Stamminger, M. 2006. Splatting indirect illumination. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*. I3D '06.

Dachsbacher, C., Stamminger, M., Drettakis, G., and Durand, F. 2007. Implicit visibility and antiradiance for interactive global illumination. In *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07.

Dammertz, H., Keller, A., and Lensch, H. P. A. 2010. Progressive point-light-based global illumination. *Computer Graphics Forum 29,* 8, 2504–2515.

Davidovič, T., Křivánek, J., Hašan, M., Slusallek, P., and Bala, K. 2010. Combining global and local virtual lights for detailed glossy illumination. In *ACM SIGGRAPH Asia 2010 papers*. SIGGRAPH ASIA '10.

Dutre, P., Bala, K., Bekaert, P., and Shirley, P. 2006. *Advanced Global Illumination*. AK Peters Ltd.

Engelhardt, T., Novák, J., Schmidt, T.-W., and Dachsbacher, C. 2012. Approximate bias compensation for rendering scenes with heterogeneous participating media. *Computer Graphics Forum (Proceedings of Pacific Graphics 2012) 31,* 7, 2145–2154.

Fischler, M. A. and Bolles, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24,* 6 (June), 381–395.

Georgiev, I., Křivánek, J., Davidovič, T., and Slusallek, P. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph. 31,* 6 (Nov.), 192:1–192:10.

Georgiev, I., Křivánek, J., Popov, S., and Slusallek, P. 2012. Importance caching for complex illumination. *Computer Graphics Forum 31,* 2. EUROGRAPHICS 2012.

Georgiev, I. and Slusallek, P. 2010. Simple and robust iterative importance sampling of virtual point lights. In *Proceedings of Eurographics 2010 (short papers)*. 57–60.

Goldstein, T. and Osher, S. 2009. The split bregman method for l1-regularized problems. *SIAM J. Img. Sci. 2*, 323–343.

Gruenschloss, L., Keller, A., Premoze, S., and Raab, M. 2012. Advanced (quasi) monte carlo methods for image synthesis. In *ACM SIGGRAPH 2012 Courses*. SIGGRAPH '12.

Gupta, M., Agrawal, A., Veeraraghavan, A., and Narasimhan, S. 2013. A practical approach to 3d scanning in the presence of interreflections, subsurface scattering and defocus. *International Journal of Computer Vision (IJCV) 102,* 1-3, 33–55.

Gupta, M. and Nayar, S. K. 2012. Micro phase shifting. In *Computer Vision and Pattern Recognition (CVPR)*. 813 –820.

GUPTA, M., TIAN, Y., NARASIMHAN, S., AND ZHANG, L. 2012. A combined theory of defocused illumination and global light transport. *International Journal of Computer Vision (IJCV) 98*, 146–167.

HACHISUKA, T., JAROSZ, W., WEISTROFFER, R. P., DALE, K., HUMPHREYS, G., ZWICKER, M., AND JENSEN, H. W. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. In *ACM SIGGRAPH 2008 Papers*. SIGGRAPH '08.

HACHISUKA, T. AND JENSEN, H. W. 2009. Stochastic progressive photon mapping. In *ACM SIGGRAPH Asia 2009 Papers*. SIGGRAPH Asia '09.

HACHISUKA, T., OGAKI, S., AND JENSEN, H. W. 2008. Progressive photon mapping. In *ACM SIGGRAPH Asia 2008 Papers*. SIGGRAPH Asia '08.

HACHISUKA, T., PANTALEONI, J., AND JENSEN, H. W. 2012. A path space extension for robust light transport simulation. *ACM Trans. Graph. 31,* 6 (Nov.), 191:1–191:10.

HARDY, J. 2012. Country kitchen - cycles - blender 2.62.

HAŠAN, M., KŘIVÁNEK, J., WALTER, B., AND BALA, K. 2009. Virtual spherical lights for many-light rendering of glossy scenes. In *ACM SIGGRAPH Asia 2009 papers*. SIGGRAPH Asia '09.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2007. Matrix row-column sampling for the many-light problem. In *ACM SIGGRAPH 2007 papers*. SIGGRAPH '07.

HEY, H. AND PURGATHOFER, W. 2002. Importance sampling with hemispherical particle footprints. In *Proceedings of the 18th Spring Conference on Computer Graphics*. SCCG '02. ACM, New York, NY, USA, 107–114.

HOLROYD, M., LAWRENCE, J., AND ZICKLER, T. 2010. A coaxial optical scanner for synchronous acquisition of 3d geometry and surface reflectance. *ACM Trans. Graph.*.

INOSHITA, C., MUKAIGAWA, Y., MATSUSHITA, Y., AND YAGI, Y. 2012. Shape from single scattering for translucent objects. In *European Conference on Computer Vision (ECCV)*.

IWAHORI, Y., SUGIE, H., AND ISHII, N. 1990. Reconstructing shape from shading images under point light source illumination. In *International Conference on Pattern Recognition (ICPR)*. Vol. 1. 83 –87.

JAKOB, W. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

JAKOB, W. AND MARSCHNER, S. 2012. Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. 31,* 4 (July), 58:1–58:13.

JENSEN, H. W. 1995. Importance driven path tracing using the photon map. In *in Eurographics Rendering Workshop*. 326–335.

JENSEN, H. W. 1996. Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*. 21–30.

KAJIYA, J. T. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '86. 143–150.

KAPLANYAN, A. AND DACHSBACHER, C. 2010. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10.

KAPLANYAN, A. AND DACHSBACHER, C. 2013a. Path space regularization for holistic and robust light transport. *Computer Graphics Forum 32,* 2, 63–72.

KAPLANYAN, A. S. AND DACHSBACHER, C. 2013b. Adaptive progressive photon mapping. *ACM Trans. Graph. 32,* 2 (apr), 16:1–16:13.

KELLER, A. 1997. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '97. 49–56.

KIM, B. AND BURGER, P. 1991. Depth and shape from shading using the photometric stereo method. *CVGIP: Image Understanding 54,* 3, 416 – 427.

KNAUS, C. AND ZWICKER, M. 2011. Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph. 30,* 3 (May), 25:1–25:13.

KOLLIG, T. AND KELLER, A. 2006. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*. Springer Berlin Heidelberg, 245–257.

KŘIVÁNEK, J., FERWERDA, J. A., AND BALA, K. 2010. Effects of global illumination approximations on material appearance. In *ACM SIGGRAPH 2010 papers*. SIGGRAPH '10.

LAFORTUNE, E. P. AND WILLEMS, Y. D. 1994. Using the modified phong reflectance model for physically based rendering. Tech. rep., Department of Computer Science, K.U.Leuven.

LEHTINEN, J., KARRAS, T., LAINE, S., AITTALA, M., DURAND, F., AND AILA, T. 2013. Gradient-domain metropolis light transport. *ACM Trans. Graph. 32,* 4.

LI, T.-M., WU, Y.-T., AND CHUANG, Y.-Y. 2012. Sure-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH Asia 2012) 31,* 6 (November), 186:1–186:9.

LIU, S., NG, T.-T., AND MATSUSHITA, Y. 2010. Shape from second-bounce of light transport. In *European Conference on Computer Vision (ECCV)*.

LOOS, B. J., ANTANI, L., MITCHELL, K., NOWROUZEZAHRAI, D., JAROSZ, W., AND SLOAN, P.-P. 2011. Modular radiance transfer. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. SIGGRAPH Asia '11.

MARA, M., LUEBKE, D., AND MCGUIRE, M. 2013. Toward practical real-time photon mapping: Efficient gpu density estimation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '13.

MCGUIRE, M. 2011. Computer graphics archive.

MERL. 2006. Merl brdf database. http://www.merl.com/brdf/.

MORÉ, J. J. 1978. The levenberg-marquardt algorithm: Implementation and theory. In *Numerical Analysis*, G. Watson, Ed. Lecture Notes in Mathematics, vol. 630. Springer Berlin Heidelberg, 105–116.

NAYAR, S. K., IKEUCHI, K., AND KANADE, T. 1991. Shape from interreflections. *International Journal of Computer Vision (IJCV) 6*, 173–195.

NAYAR, S. K., KRISHNAN, G., GROSSBERG, M. D., AND RASKAR, R. 2006. Fast Separation of Direct and Global Components of a Scene using High Frequency Illumination. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2004. Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph. 23*, 477–487.

NICHOLS, G. AND WYMAN, C. 2009. Multiresolution splatting for indirect illumination. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*. I3D '09.

NOVÁK, J., ENGELHARDT, T., AND DACHSBACHER, C. 2011. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Symposium on Interactive 3D Graphics and Games*. I3D '11.

NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., AND JAROSZ, W. 2012a. Progressive virtual beam lights. *Computer Graphics Forum (Proceedings of EGSR 2012) 31,* 4 (July).

NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., AND JAROSZ, W. 2012b. Virtual ray lights for rendering scenes with participating media. *ACM Trans. Graph. 31,* 4 (July), 60:1–60:11.

O'TOOLE, M. AND KUTULAKOS, K. N. 2010. Optical computing for fast light transport analysis. In *SIGGRAPH Asia*.

OU, J. AND PELLACINI, F. 2011. Lightslice: matrix slice sampling for the many-lights problem. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. SIGGRAPH Asia '11.

PEERS, P., MAHAJAN, D. K., LAMOND, B., GHOSH, A., MATUSIK, W., RAMAMOORTHI, R., AND DEBEVEC, P. 2009. Compressive light transport sensing. *ACM Trans. Graph. 28,* 1, 1–18.

PHONG, B. T. 1975. Illumination for computer generated pictures. *Commun. ACM 18,* 6 (June), 311–317.

Popov, S., Georgiev, I., Slusallek, P., and Dachsbacher, C. 2013. Adaptive quantization visibility caching. *Computer Graphics Forum 32,* 2, 399–408.

Ritschel, T., Engelhardt, T., Grosch, T., Seidel, H.-P., Kautz, J., and Dachsbacher, C. 2009. Micro-rendering for scalable, parallel final gathering. In *ACM SIGGRAPH Asia 2009 Papers*. SIGGRAPH Asia '09.

Ritschel, T., Grosch, T., Kim, M. H., Seidel, H.-P., Dachsbacher, C., and Kautz, J. 2008. Imperfect shadow maps for efficient computation of indirect illumination. In *ACM SIGGRAPH Asia 2008 Papers*. SIGGRAPH Asia '08.

Ritschel, T., Grosch, T., and Seidel, H.-P. 2009. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*. I3D '09.

Schechner, Y. Y., Nayar, S. K., and Belhumeur, P. N. 2003. A Theory of Multiplexed Illumination. In *IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 808–815.

Schlick, C. 1994. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum 13*, 233–246.

Segovia, B., Iehl, J. C., Mitanchey, R., and Péroche, B. 2006. Bidirectional instant radiosity. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*. EGSR'06. 389–397.

Seitz, S. M., Matsushita, Y., and Kutulakos, K. N. 2005. A theory of inverse light transport. In *International Conference on Computer Vision (ICCV)*.

Sen, P., Chen, B., Garg, G., Marschner, S. R., Horowitz, M., Levoy, M., and Lensch, H. P. A. 2005. Dual photography. In *ACM SIGGRAPH*. 745–755.

Sen, P. and Darabi, S. 2009. Compressive dual photography. *Computer Graphics Forum 28,* 2, 609–618.

Shirley, P. and Chiu, K. 1997. A low distortion map between disk and square. *J. Graph. Tools 2,* 3 (Dec.), 45–52.

Shirley, P., Wang, C., and Zimmerman, K. 1996. Monte carlo techniques for direct lighting calculations. *ACM Trans. Graph. 15,* 1 (Jan.), 1–36.

Talbot, J. F., Cline, D., and Egbert, P. 2005. Importance resampling for global illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*. EGSR '05. 139–146.

Tokuyoshi, Y. and Ogaki, S. 2012. Real-time bidirectional path tracing via rasterization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '12.

VEACH, E. 1998. Robust monte carlo methods for light transport simulation. Ph.D. thesis, Stanford, CA, USA.

VEACH, E. AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. 65–76.

VORBA, J., KARLÍK, O., ŠIK, M., RITSCHEL, T., AND KŘIVÁNEK, J. 2014. On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph. 33,* 4 (July), 101:1–101:11.

WALTER, B. 2005. Notes on the ward brdf. Tech. rep., Program of Computer Graphics, Cornell University. April.

WALTER, B., ARBREE, A., BALA, K., AND GREENBERG, D. P. 2006. Multidimensional lightcuts. In *ACM SIGGRAPH 2006 Papers*. SIGGRAPH '06.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: A scalable approach to illumination. In *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05.

WALTER, B., KHUNGURN, P., AND BALA, K. 2012. Bidirectional lightcuts. *ACM Trans. Graph. 31,* 4 (July), 59:1–59:11.

WANG, C. A. 1994. The direct lighting computation in global illumination methods. Ph.D. thesis, Bloomington, IN, USA.

WANG, R. AND ÅKERLUND, O. 2009. Bidirectional importance sampling for unstructured direct illumination. *Computer Graphics Forum 28,* 2, 269–278.

WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13,* 4 (apr), 600–612.

WARD, G. J. 1992. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '92. 265–272.

WAYNE, W. 2014. The breakfast room - cycles - blender 2.71.

WU, Y.-T. AND CHUANG, Y.-Y. 2013. Visibilitycluster: Average directional visibility for many-light rendering. *IEEE Transactions on Visualization and Computer Graphics 19,* 9 (Sept.), 1566–1578.

YAMAZAKI, S., MOCHIMARU, M., AND KANADE, T. 2011. Simultaneous self-calibration of a projector and a camera using structured light. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*. 60 –67.

Yin, W., Morgan, S., Yang, J., and Zhang, Y. 2010. Practical compressive sensing with toeplitz and circulant matrices. *Proceedings of Visual Communications and Image Processing (VCIP).*

Yoon, K.-J., Prados, E., and Sturm, P. 2010. Joint estimation of shape and reflectance using multiple images with known illumination conditions. *International Journal of Computer Vision (IJCV) 86,* 2-3 (January), 192–210.

Appendix A

More implementation details

A.1 Probability density function

A.1.1 Changing variables in probability density function

The probability $p(\omega)$ and $p(\theta, \phi)$ can be related by

$$p(\omega)\mathrm{d}\omega = (p(\omega)\sin\theta)\mathrm{d}\theta\mathrm{d}\phi = p(\theta,\phi)\mathrm{d}\theta\mathrm{d}\phi, \tag{A.1}$$

which leads to

$$p(\theta,\phi) = p(\omega)\sin\theta. \tag{A.2}$$

A.1.2 Deriving cosine-weighted sampling formula

The marginal probability $p(\theta)$ is

$$p(\theta) = \int_0^{2\pi} p(\theta,\phi)\mathrm{d}\phi = \sin 2\theta. \tag{A.3}$$

In order to sample θ from a uniform variable δ_1, we simply let the cumulative density function $F(\theta)$ be equal to δ_1:

$$F(\theta) = \int_0^{\theta} p(\theta')\mathrm{d}\theta' = \sin^2\theta = \delta_1. \tag{A.4}$$

From that we obtain

$$\theta = \sin^{-1}(\sqrt{\delta_1}). \tag{A.5}$$

Given θ, we now proceed to sample ϕ. We have

$$p(\phi \mid \theta) = \frac{p(\theta,\phi)}{p(\theta)} = \frac{1}{2\pi}. \tag{A.6}$$

The cumulative density $F(\phi \mid \theta)$ can be easily derived:

$$F(\phi \mid \theta) = \int_0^{\phi} p(\phi' \mid \theta)\mathrm{d}\phi' = \frac{\phi}{2\pi} = \delta_2. \tag{A.7}$$

Therefore, we have

$$\phi = 2\pi\delta_2. \tag{A.8}$$

A.2 Form factor

In global illumination algorithms, form factor is an important mathematical term to compute light bounces among surfaces. Form factor represents the fraction of power from a patch at $\mathbf{y}$ to a patch at $\mathbf{x}$ and can be written as

$$F(\mathbf{y}, \mathbf{x}) = \frac{1}{A_{\mathbf{x}}} \int_{A_{\mathbf{x}}} \int_{A_{\mathbf{y}}} \frac{G(\mathbf{y}, \mathbf{x})V(\mathbf{y}, \mathbf{x})}{\pi} \mathrm{d}A(\mathbf{y})\mathrm{d}A(\mathbf{x}). \tag{A.9}$$

The form factor can be interpreted as the power from $\mathbf{y}$ to $\mathbf{x}$ per unit surface area at $\mathbf{x}$ on average. Note that

$$A_{\mathbf{x}}F(\mathbf{y}, \mathbf{x}) = A_{\mathbf{y}}F(\mathbf{x}, \mathbf{y}), \tag{A.10}$$

which suggests the relation between the form factor from $\mathbf{y}$ to $\mathbf{x}$ and from $\mathbf{x}$ to $\mathbf{y}$, where $A_{\mathbf{x}}$ and $A_{\mathbf{y}}$ are areas of patch at $\mathbf{x}$ and $\mathbf{y}$, respectively. This relation is used in progressive radiosity [Cohen et al. 1988] to centralize form factor computation from a patch $\mathbf{x}$ to all other patches at each iteration.

Form factor can also be approximated by

$$F(\mathbf{y}, \mathbf{x}) = \int_{A_{\mathbf{y}}} \frac{G(\mathbf{y}, \mathbf{x})V(\mathbf{y}, \mathbf{x})}{\pi} \mathrm{d}A(\mathbf{y}), \tag{A.11}$$

if the patch $\mathbf{x}$ is very small. If patch $\mathbf{y}$ is also small and far away, we can also approximate

$$F(\mathbf{y}, \mathbf{x}) = \frac{G(\mathbf{y}, \mathbf{x})V(\mathbf{y}, \mathbf{x})}{\pi} A(\mathbf{y}). \tag{A.12}$$

Form factor is computed numerically. Only form factors for special configurations such as two parallel planes or a pair of perpendicular planes have closed-form formulas (see [Dutre et al. 2006], page 215). In Monte Carlo estimation, form factor appears as the geometry term G. The sampling of areas and solid angles 'hides' the area term in the form factor into the probability term.

In Chapter 7, the form factor is further analyzed for geometry reconstruction.

A.3 Conversion between VPL and photon

A.3.1 Reflected radiance using photons

In photon mapping, we store at each particle the flux of each photon while in VPL rendering, we often store the incident radiance from the previous virtual point light. Note that flux is defined for a certain point while incident radiance is defined from point $\mathbf{y}$ to point $\mathbf{x}$. As stated by [Hachisuka et al. 2012; Georgiev et al. 2012], this is the one-bounce difference between photon mapping and bidirectional path tracing.

Suppose that we need to evaluate the outgoing radiance $L_o(\mathbf{x} \to \omega)$. Photon mapping uses the photons in the local neighborhood of $\mathbf{x}$ to approximate the incident flux to $\mathbf{x}$. To convert the incident photon flux to reflected radiance, Jensen [1996] proposed the following formula:

$$L_o(\mathbf{x} \to \omega) = \sum_{\mathbf{y}} \frac{\Phi_{\mathbf{y}}}{\mathrm{d}A_{\mathbf{x}}} f_s(\omega_i(\mathbf{y}) \to \mathbf{x} \to \omega), \tag{A.13}$$

where $\mathrm{d}A_{\mathbf{x}} = \pi r^2$, r the radius of the sphere centered at $\mathbf{x}$ that contains the nearest N photons, and $\omega_i(\mathbf{y})$ the incident direction that the photon $\mathbf{y}$ receives flux from its previous photon. In other words, the area $\mathrm{d}A_{\mathbf{x}}$ is approximated by the disk intersected by the sphere and the surface that contain $\mathbf{x}$.

A.3.2 Reflected radiance using VPLs

A virtual point light stores the throughput and the probability of the light path that it represents. Given a VPL at $\mathbf{y}$, the reflected radiance at a surface point $\mathbf{x}$ due to $\mathbf{y}$ can be calculated by

$$\begin{aligned} L_o(\mathbf{x}, \omega) &= L_i(\mathbf{y} \to \mathbf{x}) G(\mathbf{y}, \mathbf{x}) f_s(\omega_i(\mathbf{y}) \to \mathbf{x} \to \omega) \\ &= \frac{T(\bar{y})}{p(\bar{y})} f_s(\omega_i(\mathbf{y}) \to \mathbf{y} \to \mathbf{x}) G(\mathbf{y}, \mathbf{x}) f_s(\mathbf{y} \to \mathbf{x} \to \omega), \end{aligned} \tag{A.14}$$

where $T(\bar{y})$ is the throughput of the light path that ends at the location $\mathbf{y}$, and $p(\bar{y})$ the probability of the path.

A.3.3 From photon to VPL

Given the incident flux $\Phi_{\mathbf{y}}$, the radiant intensity from $\mathbf{y}$ to $\mathbf{x}$ can be calculated as

$$\begin{aligned} I(\mathbf{y} \to \mathbf{x}) &= L_o(y \to \mathbf{x}) A_{\mathbf{y}} \cos\theta_{\mathbf{y}} \\ &= \left(L_i(\mathbf{z} \to \mathbf{y}) f_s(\mathbf{z} \to \mathbf{y} \to \mathbf{x}) \cos\delta_{\mathbf{y}} \Omega_{\mathbf{z}}\right) A_y \cos\theta_{\mathbf{y}} \\ &= \left(L_i(\mathbf{z} \to \mathbf{y}) A_y \cos\delta_{\mathbf{y}} \Omega_{\mathbf{z}}\right) f_s(\mathbf{z} \to \mathbf{y} \to \mathbf{x}) \cos\theta_{\mathbf{y}} \\ &= \Phi_y f_s(\mathbf{z} \to \mathbf{y} \to \mathbf{x}) \cos\theta_{\mathbf{y}}, \end{aligned} \tag{A.15}$$

where θ and δ denote the angle to the surface normal of the incident and outgoing ray, respectively. Therefore, if we consider the photon as a VPL, the reflected radiance at $\mathbf{x}$ due to the photon can be derived as follows.

$$\begin{aligned} L_o(\mathbf{x} \to \omega) &= L_i(\mathbf{y} \to \mathbf{x}) f_s(\mathbf{y} \to \mathbf{x} \to \omega) \cos\delta_{\mathbf{x}} \Omega_{\mathbf{y}} \\ &= L_i(\mathbf{y} \to \mathbf{x}) f_s(\mathbf{y} \to \mathbf{x} \to \omega) \cos\delta_{\mathbf{x}} \frac{A_{\mathbf{y}} \cos\theta_{\mathbf{y}}}{\|\mathbf{y} - \mathbf{x}\|^2} \\ &= \left(L_o(\mathbf{y} \to \mathbf{x}) A_{\mathbf{y}} \cos\theta_y\right) f_s(\mathbf{y} \to \mathbf{x} \to \omega) \cos\delta_{\mathbf{x}} / \|\mathbf{y} - \mathbf{x}\|^2 \\ &= I(\mathbf{y} \to \mathbf{x}) f_s(\mathbf{y} \to \mathbf{x} \to \omega) \cos\delta_{\mathbf{x}} / \|\mathbf{y} - \mathbf{x}\|^2 \\ &= \Phi_y f_s(\mathbf{z} \to \mathbf{y} \to \mathbf{x}) \frac{\cos\theta_{\mathbf{y}} \cos\delta_{\mathbf{x}}}{\|\mathbf{y} - \mathbf{x}\|^2} f_s(\mathbf{y} \to \mathbf{x} \to \omega) \\ &= \Phi_y f_s(\mathbf{z} \to \mathbf{y} \to \mathbf{x}) G_{\mathbf{yx}} f_s(\mathbf{y} \to \mathbf{x} \to \omega). \end{aligned} \tag{A.16}$$

A.3.4 From VPL to photon

In order to use a VPL as a photon, it is necessary to evaluate the incident flux at a VPL. We provide a simple derivation as follows. Suppose that the photon $\mathbf{y}$ represents a surface with area $A_{\mathbf{y}}$ and it receives the flux from another surface with area $A_{\mathbf{z}}$. The incident flux to $\mathbf{y}$ can be written as

$$\begin{aligned} \Phi_{\mathbf{y}} &= \int_{A_y} \int_{A_z} L_i(\mathbf{z} \to \mathbf{y}) \frac{\cos\delta_{\mathbf{y}} \cos\theta_{\mathbf{z}}}{\|\mathbf{z} - \mathbf{y}\|^2} \mathrm{d}A_{\mathbf{y}} \mathrm{d}A\mathbf{z} \\ &= \frac{L_i(\mathbf{z} \to \mathbf{y}) G_{\mathbf{zy}}}{p(\mathbf{z}) p(\mathbf{y})}. \end{aligned} \tag{A.17}$$

By expanding the incident radiance $L_i(\mathbf{z} \to \mathbf{y})$ recursively towards the light source, we obtain a general formula to approximate the incident flux as

$$\Phi_{\mathbf{y}} = \frac{T(\bar{\mathbf{y}})}{p(\bar{\mathbf{y}})}. \tag{A.18}$$

Therefore, the conversion from a virtual point light to a photon is straightforward.

A.4 Hemispherical mapping

To map the unit hemisphere to the unit square, we first map the unit hemisphere to the unit disk such that a uniform distribution on the hemisphere becomes a uniform distribution on the disk:

$$\begin{aligned} x &= u\sqrt{2 - w^2}, \\ y &= v\sqrt{2 - w^2}, \\ z &= 1 - w^2, \end{aligned} \tag{A.19}$$

where $w^2 = u^2 + v^2$, (x, y, z) and (u, v) are points on the unit hemisphere and the unit disk, respectively. Points on the unit disk can then be mapped to the unit square using concentric mapping [Shirley and Chiu 1997]. After this mapping, incoming radiance estimation for a direction in the unit hemisphere can be cast to estimation for a point in the unit square. We have $\mathrm{d}\omega = 2\pi\mathrm{d}s$, or $p(\omega) = p(s)/2\pi$, where ω is a point on the unit hemisphere, and s a point in the unit square. The constant factor 2π would be necessary when an integral defined in the unit hemisphere domain is estimated in the unit square domain.