Department of Computer Science
University of Pretoria
Pretoria
South Africa

Light Beam Tracing for Multi-Bounce Specular and Glossy Transport Paths

by

Bernardt Duvenhage

May 15, 2015

Submitted in partial fulfilment of the requirements for the degree Doctor of Philosophy in Science (Computer Science) in the Faculty of Engineering, Built Environment and Information Technology

University of Pretoria
Pretoria, South Africa

Superviser:	Professor D. G. Kourie
Co-Superviser:	Professor K. Bouatouch (University of Rennes 1, France)

Acknowledgements

I firstly wish to acknowledge the support that my employer, the Council for Scientific and Industrial Research (CSIR), has given me for completing this thesis. I would like to express my sincerest gratitude to Professor Derrick Kourie and Professor Kadi Bouatouch for their expert guidance during my PhD and for their very valuable contributions as co-authors of my conference and journal papers. I would also like to thank my lovely family for their support and for giving me the time to write this thesis. Finally, I would like to thank the computer graphics community for making the field so stimulating and challenging.

Abstract

Light beam tracing is an efficient rendering algorithm for simulating caustics, the envelopes of light that are scattered from shiny curved surfaces *and* focussed into lines or spots of concentrated light. Light beam tracing is efficient for rendering caustics because the algorithm is able to exploit the coherency of the transport paths within an envelope of light. However, light beam tracing rendering algorithms found in the literature only support mirror-like specular surface interactions. Therefore, there is motive for extending light beam tracing to include more realistic roughened specular and other glossy surfaces while maintaining the efficiency of the rendering algorithm.

This thesis first offers a conjecture on how to extend light beam tracing to include glossy surface interactions. The glossy bidirectional reflectance distribution function (BRDF) that is required to support the conjecture is then derived and shown to be physically plausible. Following from the conjecture a new extension to light beam tracing that allows glossy surface interactions for more realistic rendering of caustics is formulated.

Gauss' divergence theorem is used to replace the irradiance surface integral of the lighting equation with a more efficient boundary line integral. This solution is also shown to be reusable for all-frequency interactions although more work is required to complete the derivations.

Finally, multi-bounce glossy light beam tracing is demonstrated which further extends the application domain of glossy light beam tracing. The new rendering algorithm is shown to be a good alternative for rendering single-bounce and multi-bounce caustics due to specular as well as glossy surfaces. The expectation is that the irradiance solution would also in future be useful for more general applications.

Contents

I Introduction, Background and Related Work 11

1 Introduction 15
1.1 Overview . . . 16
1.2 Brief Scope of Work . . . 17
1.3 Structure and Layout . . . 17
1.4 Summary of Contributions . . . 18

2 Background 19
2.1 Realistic Image Synthesis . . . 19
2.2 Radiometry . . . 20
2.3 The Rendering Equation . . . 26
2.4 Spherical Gaussian Functions . . . 29
2.5 The Problem and Thesis Scope . . . 30

3 Related Work 33
3.1 Early Work . . . 33
3.2 Light Beam Tracing and Caustics . . . 36
3.3 Spherical Gaussians and Caustics . . . 40
3.4 Summary . . . 41

II The Glossy Light Beam Tracing Conjecture 43

4 Classical Backward Beam Tracing 47
4.1 Overview . . . 47
4.2 Tracing Light Beams . . . 48

4.3 The Irradiance Estimate 51

5 Conjecture **55**
5.1 Overview 55
5.2 The Glossy Irradiance Estimate 56

6 Results and Analysis **65**
6.1 Results 66
6.2 Analysis 69

7 Summary **73**

III The Glossy Scatter Lobe BRDF **75**

8 Derivation of the Glossy BRDF **79**
8.1 Overview 79
8.2 Derivation 80
8.3 Physical Plausibility 82
8.3.1 Positivity 82
8.3.2 Symmetricity 83
8.3.3 Conservation of Energy 83

9 Numerical Verification of the BRDF **85**
9.1 Generating Equidistant Input Vectors 87
9.2 Verifying the Random Vector PDFs 91
9.3 Verifying the BRDF Against Its Photon Scatter Operators . . 93
9.4 Verifying BRDF Symmetry 95
9.5 Verifying BRDF Energy Conservation 96

10 Results and Analysis **97**
10.1 Results 97
10.2 The Lambertian Diffuse BRDFs 98
10.3 The Blinn-Phong BRDF 101
10.4 The Phong BRDF 103

10.5 The Glossy BRDF 105
10.6 Analysis 107

11 Summary **111**

IV The Glossy Light Beam Tracing Algorithm **113**

12 The Glossy Irradiance Estimate **117**
12.1 Overview 117
12.2 Derivation 119
12.3 Using Gauss' Theorem 123
12.4 Shadowing 128
12.5 All-Frequency Surface Interactions 129

13 Multi-bounce Beam Tracing **133**
13.1 Overview 134
13.2 Derivation 136

14 Results and Analysis **141**
14.1 Results 142
14.2 Analysis 144

15 Summary **149**

V About the Code **151**

16 StitchEngine Overview **155**
16.1 Call Graphs and Class Hierarchies 156
16.2 Optimisation Strategies 160
16.3 The Bounding Volume Hierarchies 162

17 The Renderer Implementations **163**
17.1 The Light Trace Renderer 163
17.2 The Light Beam Renderer 164

17.3 The Photon Map Renderer 164

VI Conclusion and Future Work 167

18 Conclusion 171

19 Future Work 175

20 Poster 177

Glossary 179

Acronyms 183

Bibliography 183

Part I

Introduction, Background and Related Work

This first part of the thesis gives the introduction, background for and related works to the thesis topic of light beam tracing (LBT).

Chapter 1

Introduction

Caustics, shown in Figure 1.1, produce important cues for vision and scene understanding. This thesis is on the topic of synthesising and rendering of these light transport paths using light beam tracing (LBT).

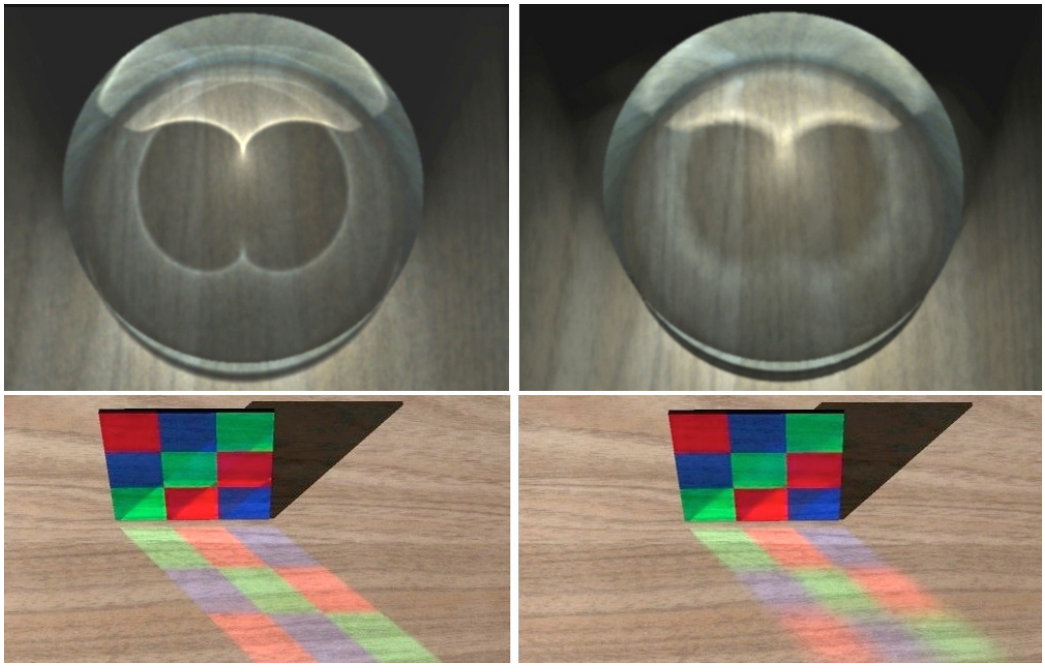


Figure 1.1: Caustics due to light reflecting off a mirror-like surface (left) and off a roughened specular or glossy surface (right).

1.1 Overview

Using Heckbert's [1] regular expression notation[1] for light transport paths, the light paths that typically produce the strongest caustics may be expressed as $L(S|G)^*D$.

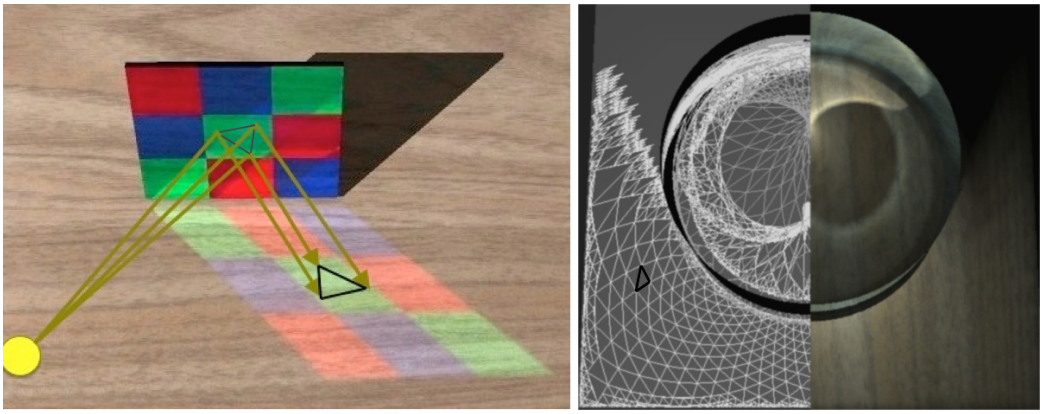


Figure 1.2: Left: Shows a single light beam (bounded by three light paths) reflected off a specular surface. Right: The refined light beams of a cardioid caustic.

The physics of light transport have been brought together in Kajiya's [2] rendering equation. The algorithms that have been designed to solve the rendering equation are known as *rendering algorithms*. These algorithms usually numerically solve the multi-dimensional integral rendering equation. Points, lines, or more complex primitives, such as the light beams shown in Figure 1.2, are used to sample the set of all light transport paths which is the domain of integration of the rendering equation.

The terms *backward rendering* and *backward tracing* are used interchangeably with light tracing to describe ray or beam tracing *from the light* in the same direction as the flow of radiance. Forward tracing and forward rendering is used to describe ray or beam tracing from the camera in the same direction as the flow of importance[2] [3]. This convention ties up with the related works such as backward ray tracing [4] and backward beam tracing [5]

[1] Specular, glossy and diffuse material interactions are denoted by S, G and D respectively, while the light source and camera are denoted by L and E respectively.

[2] Importance is the adjoint of radiance as an importon is the adjoint of a photon.

(BBT) described in Chapter 3.

LBT rendering algorithms have in the past been used to effectively lump[3] together neighbouring multi-bounce *specular* transport paths (LS^*D). Tracing light beams instead of individual rays reduces the number of transport operations needed to render a scene and therefore could result in more efficient rendering. The complexity of tracing and evaluating a light beam is, however, higher than that of tracing a light photon or light ray. Herein lies the challenge in implementing LBT.

1.2 Brief Scope of Work

This thesis describes an extension of LBT to also include glossy surface interactions. The scope of the work encompasses both the derivation of the new physically based rendering algorithm as well as a multi-core CPU implementation of it and of other reference rendering algorithms.

The software that was written for the thesis and used to generate the results is available at: `http://code.google.com/p/stitch-engine/source` tag sprinkles[4].

1.3 Structure and Layout

This first part of the thesis gives the introduction, background and related works to the thesis topic.

The next part, Part II, presents a conjecture[5] on an irradiance estimate that is expected to lead to a LBT rendering algorithm that supports glossy surfaces. Following the scientific method the proposed irradiance estimate is evaluated so that if found plausible the rendering algorithm may be developed further.

Part III describes the derivation and verification for physical plausibility

[3] A lumped element model simplifies the behaviour of a physical system under certain assumptions to a single component or equation.

[4] The motivation for the tag name is given in `CoffeeShake.txt` in the code repository.

[5] A conjecture is taken to mean an unproven proposition that appears correct.

of the glossy BRDF that is required to support the conjecture given in the antecedent, Part II. Verified physical plausible BRDF *implementations* are also a critical requirement for comparing different rendering algorithms.

Based on the positive results of Part II and Part III the next part, Part IV then formally derives the proposed irradiance estimate and further develops the single- and multi-bounce glossy LBT algorithms.

Part V discusses some important parts of the software code that accompanies the thesis and which was developed as part of the PhD study.

Part VI critically reflects on the success of the undertaking, highlights limitations and points ahead to future work. The study into light beam tracing and the proposed light beam radiance estimate should in no way conclude with this thesis.

I have also prepared a poster on the thesis which provides a one page overview of the research. I have previously received positive feedback when presenting such an overview and include the poster in Chapter 20.

1.4 Summary of Contributions

This thesis makes several contributions:

- A new spherical Gaussian glossy BRDF is formulated and verified for physical plausibility.
- Single bounce LBT is extended to include glossy surface interactions, thus broadening the application domain of LBT.
- Gauss' divergence theorem is used to replace the surface integral of the rendering equation with a boundary line integral. This enhances[6] both the *quality and performance* of single bounce glossy LBT. As far as I know this is the first application of Gauss' divergence theorem to solve the lighting integral for rendering caustics.
- A light image is used to extend glossy LBT to *multi-bounce glossy* transport paths.

[6] One might tongue-in-cheek say "Stokes".

Chapter 2

Background

Realistic image synthesis is useful in many application domains including architecture, advertising, computer games and a whole plethora of predictive simulations. Understanding such synthesis relies on a number of domain-specific terms, as well as various concepts and assumptions. The purpose of this chapter is to briefly overview such terms, concepts and assumptions that are used in the rest of this thesis and required to illuminate the problem statement and scope of the thesis.

2.1 Realistic Image Synthesis

This thesis focusses on the second stage of Greenberg et al.'s [6] three stage framework for realistic image synthesis, namely *light transport*. Part III also makes a brief foray into the first stage, namely *measurement and modelling*.

Usually one would employ Kajiya's [2] rendering equation for realistic image synthesis. These first two stages of realistic image synthesis then simulate the physics of it all and is referred to as physically based rendering.

The geometric optics assumption is generally appropriate for physically based rendering. In terms of this assumption, light is considered to travel along straight lines as if consisting of particulate photons.

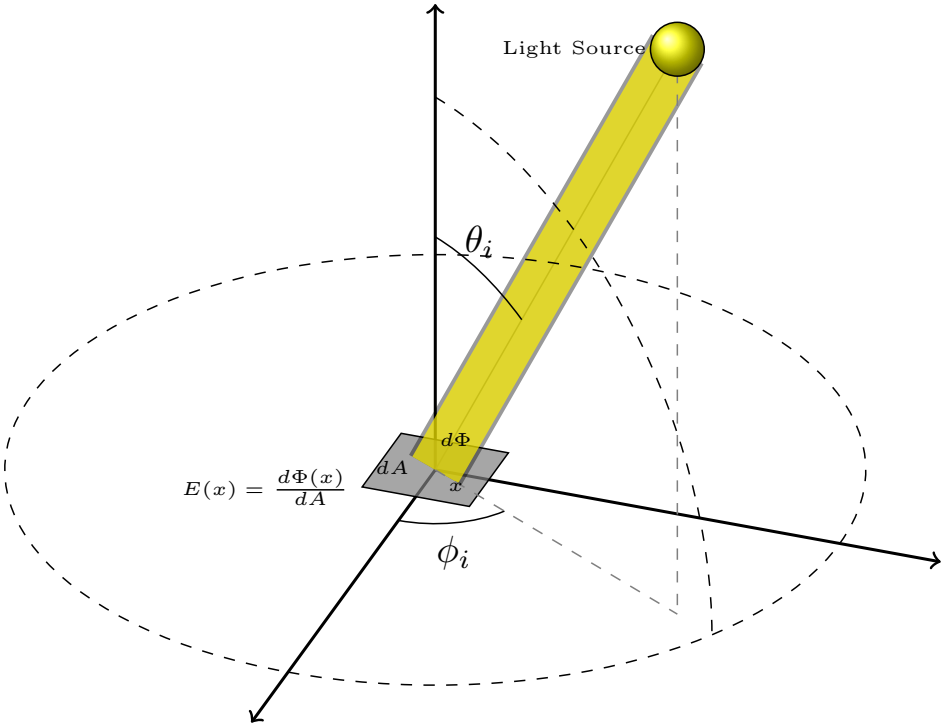


Figure 2.1: Irradiance expressed in W/m^2.

2.2 Radiometry

The radiometric quantities that are useful for physically based rendering and that are used in this thesis are flux, irradiance, radiant intensity and radiance.

- Flux, expressed in watts (W), is the amount of energy per unit time that flows through a real or virtual boundary surface. The symbol for flux is Φ. Flux could be a wide-band (grayscale) scalar value or a function of wavelength λ.

- Irradiance, $E(x)$, is the flux per unit area incident on a surface at a point x shown in Figure 2.1. It is expressed in W/m^2:

$$E\left(x\right) = \frac{d\Phi\left(x\right)}{dA} \tag{2.1}$$

  dA is often used to indicate a differential surface area and $\Phi(x)$ is

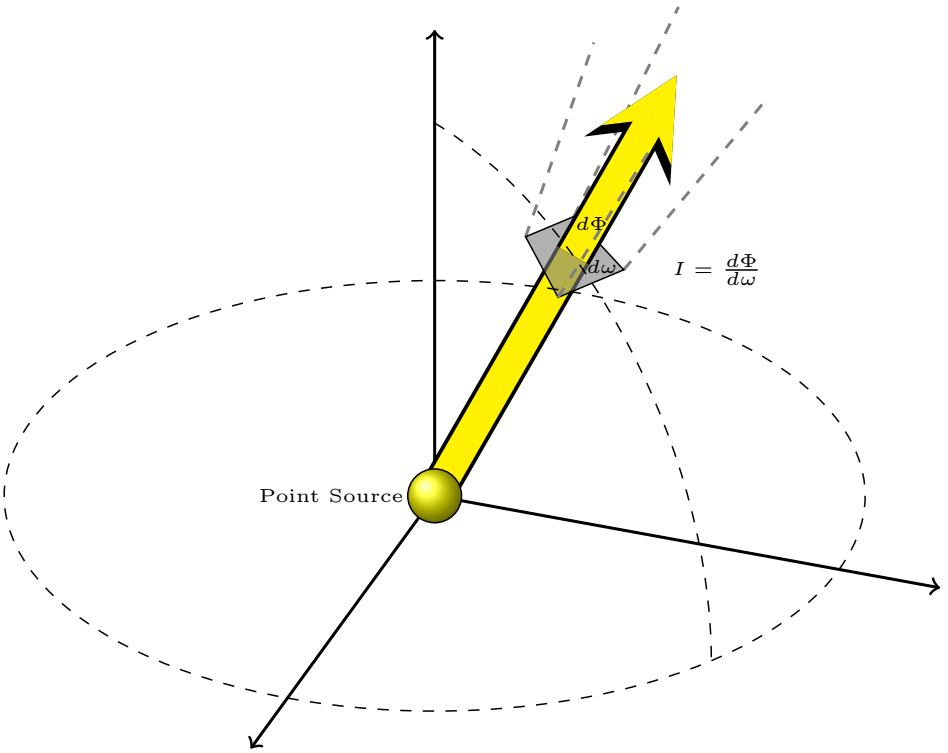


Figure 2.2: Radiant intensity expressed in W/sr.

usually written as only Φ.

- Solid angle, a two dimensional angle expressed in steradian (sr), is not itself a radiometric quantity, but it is used in the definition of radiant intensity and radiance. The solid angle subtended by an object as measured from a point x is equivalent to the area subtended by the projection of the object on a unit sphere centred on x.

- The radiant intensity, I, is the flux (also termed radiant power) per unit solid angle radiated from a point source shown in Figure 2.2. Radiant intensity is expressed in W/sr:

$$I = \frac{d\Phi}{d\omega}$$

$d\omega$ is often used to indicate a differential solid angle as shown in Fig-

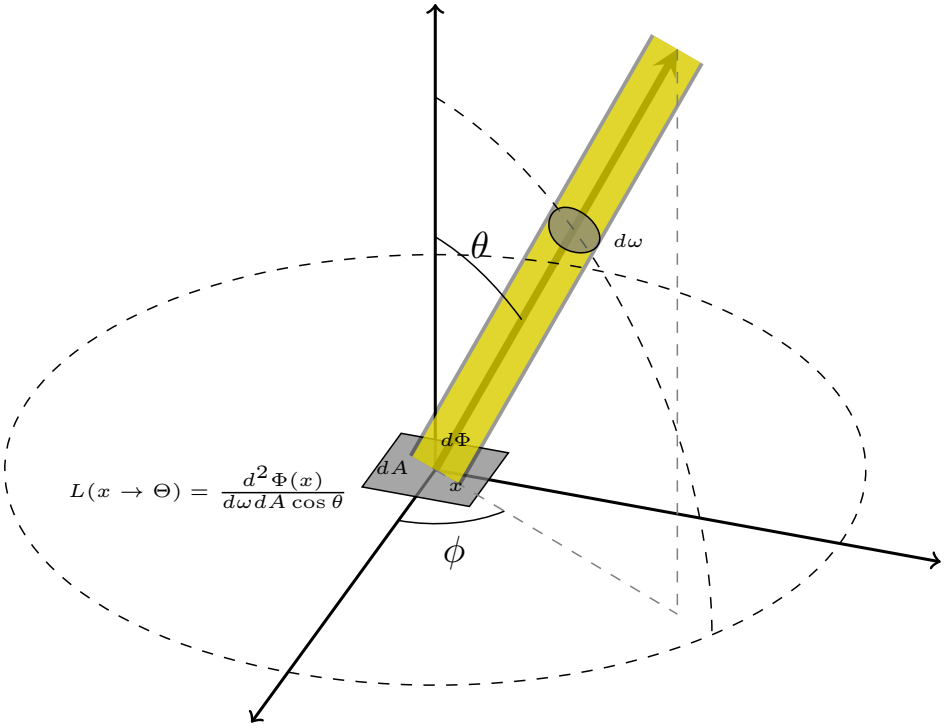


Figure 2.3: Radiance expressed in W/(sr.m^2).

ure 2.2.

- Radiance $L(x \to \Theta)$ is the flux per unit projected area per solid angle leaving or passing through x in direction Θ shown in Figure 2.3. The upper case symbols Ψ and Θ are often used to indicate a direction vector. Radiance is expressed in W/(sr . m^2):

$$L(x \to \Theta) = \frac{d^2\Phi}{d\omega dA_\perp} = \frac{d^2\Phi}{d\omega dA \cos\theta}$$

  The projected or orthogonal area $dA_\perp$ may be calculated from the surface area using $dA_\perp = d\omega dA \cos\theta$. θ is the outgoing *angle*.

In a vacuum radiance is invariant along straight paths. This invariance may be expressed as $L(x \to y) = L(y \leftarrow x)$. This states that radiance leaving from point x towards point y is the same as the radiance arriving at

point y from the direction of point x. The direction of the arrow indicates the direction of the flow of radiance.

Another attractive property of radiance is that the response of sensors such as the human eye, cameras, etc. are proportional to the at-sensor radiance of the scene. The perception of radiance is often called *brightness*.

In theory, atmospheric attenuation breaks the invariance mentioned above. However, in many real-life scenarios the atmospheric attenuation is negligible. As a consequence, the at-target radiance may be used as an approximation of the at-sensor radiance—i.e. as the radiance at the camera. Put differently, in most real-life situations the brightness of an image pixel is independent of the camera's distance to the object.

The relation between the flow of light energy onto a material and the scattered energy flow leaving the material is modelled by the bidirectional scattering distribution function (BSDF). Scattering is used to refer to the combination of reflecting and transmitting. When only reflection is considered, then the relation is modelled by the Bidirectional Reflectance Distribution Function (BRDF)[1]. The BRDF is defined in terms of incoming irradiance, E, at some point, x, and the resulting outgoing radiance, L, as:

$$\begin{aligned} f_r(x, \Psi_i \rightarrow \Psi_o) &= \frac{dL(x \rightarrow \Psi_o)}{dE(x \leftarrow \Psi_i)} \\ &= \frac{dL(x \rightarrow \Psi_o)}{L(x \leftarrow \Psi_i)|\cos(N_x, \Psi_i)|d\omega_{\Psi_i}} \end{aligned} \tag{2.2}$$

Here $dL(x \rightarrow \Psi_o)$ denotes the differential radiance reflected in direction Ψ_o. $dE(x \leftarrow \Psi_i)$ is the differential irradiance at the surface due to the incoming radiance $L(x \leftarrow \Psi_i)$ from a differential solid angle $d\omega_{\Psi_i}$ around the incoming direction Ψ_i. The uppercase symbol N is often used to indicate the average normal of a surface. N_x is the average surface normal at point x. The $|.|$ operator, used around $\cos(N_x, \Psi_i)$, takes the absolute value of the operand. It is not as in some texts a clamp to positive values.

[1] Just for interest's sake, one of the earliest definitions of the BRDF/BSDF is by Nicodemus [7]

The BRDF is often written as $f_r(x, \Psi_i \leftrightarrow \Psi_o)$ with a double arrow. The double arrow, $\leftrightarrow$, indicates that the equation holds regardless of the direction of the flow of radiance because the BRDF is symmetric. Symmetricity is a BRDF property discussed further in Chapter 8.

From the definition of the BRDF in Equation 2.2 one may calculate the differential radiance $dL(x \rightarrow \Psi_o)$ leaving point x given radiance arriving at point x:

$$dL(x \rightarrow \Psi_o) = f_r(x, \Psi_i \rightarrow \Psi_o) L(x \leftarrow \Psi_i) |\cos(N_x, \Psi_i)| d\omega_{\Psi_i} \quad (2.3)$$

The Lambertian[2] [8] diffuse surface BRDF describes an ideal matte (viz. Lambertian diffuse) surface. Lambert's cosine law states that the radiant intensity observed from a Lambertian surface is directly proportional to $|\cos(N_x, \Psi_o)|$. A consequence of Lambert's cosine law is that the radiance observed from a Lambertian surface is constant. The Lambertian diffuse BRDF is therefore constant and defined as:

$$f_r(x, \Psi_i \rightarrow \Psi_o) = \frac{\rho_d}{\pi} \quad (2.4)$$

where ρ_d is the diffuse reflectivity. To tie this back to the properties of radiance, if a *diffuse* building face illuminated by the sun is viewed by a camera then the brightness of the building in the image is independent of the distance between the building and the camera *and* independent of the viewing orientation.

Moving the discussion from diffuse to specular surfaces, the mirror-like specular surface BRDF is defined as:

$$f_r(x, \Psi_i \rightarrow \Psi_o) = \frac{\rho_s \delta(\phi)}{|\cos(N_x, \Psi_i)|} \quad (2.5)$$

where

- ρ_s is the specular reflectivity which is typically smaller than 1.0.

[2]Lambertian is named after Johann Heinrich Lambert, for his Photometria [8] book, published in 1760.

- ϕ is the angle between Ψ_o and the specular scatter direction R. Note that R may, in turn, be computed from the equation:

$$R = 2(\Psi_i \cdot N_x)N_x - \Psi_i \tag{2.6}$$

  Note that the symbol α is often used in place of ϕ mostly because ϕ is usually used to represent the azimuth angle, but in this thesis I have made use of ϕ.

- $\delta(\phi)$ is the Dirac delta function. The Dirac delta function is zero everywhere except at the origin *and* has an integral of one over its domain.

Note that the $|\cos(N_x, \Psi_i)|$ term is inherent in this and other specular BRDFs to counteract the effects of grazing incidence. If the specular BRDF is substituted into Equation 2.2 then the outgoing radiance $L(x \rightarrow \Psi_o)$ should be equal to ρ_s times the incoming radiance $L(x \rightarrow \Psi_i)$:

$$\begin{aligned}
f_r(x, \Psi_i \rightarrow \Psi_o) = \frac{dL(x \rightarrow \Psi_o)}{L(x \leftarrow \Psi_i)|\cos(N_x, \Psi_i)|d\omega_{\Psi_i}} &= \frac{\rho_s \delta(\phi)}{|\cos(N_x, \Psi_i)|} \\
dL(x \rightarrow \Psi_o) &= \rho_s L(x \leftarrow \Psi_i)\delta(\phi)d\omega_{\Psi_i} \\
L(x \rightarrow \Psi_o) &= \int_{\Omega_{\Psi_i}} \rho_s L(x \leftarrow \Psi_i)\delta(\phi)d\omega_{\Psi_i} \\
L(x \rightarrow \Psi_o) &= \rho_s L(x \leftarrow \Psi_i) \\
&\quad \text{(where } R = \Psi_o\text{)}
\end{aligned}$$

Expressions for the specular BRDF similar to Equation 2.5 are also discussed in Veach's thesis [9], and by Pharr and Humphreys [10].

The diffuse and specular material models may also be combined. One such example is the Phong [11] BRDF in which the diffuse and specular components are simply added together. To model the scattered light from specular as well as glossy surfaces[3] the Phong BRDF makes use of a $\cos^s$ probability density function (PDF), s being the specular exponent of the specular cosine scatter lobe.

[3] A glossy surface is a roughened specular surface that results in blurry reflections.

It is often necessary to find the PDF, $P_s(\Psi_o)$, of the scattered and absorbed light given a BRDF and the incoming light direction. For a specific N_x and Ψ_i, $P_s(\Psi_o) = f_r(x, \Psi_i \rightarrow \Psi_o)|\cos(N_x, \Psi_o)|$. Note that the probability volume above the surface and within this probability density function is in general smaller than one. Some light is absorbed into the surface and not scattered to the hemisphere above the surface. The scattered part of the density function has the same shape as the radiant intensity function.

Phong's original BRDF lacked physical plausibility. As will be described in Part III, equation 2.7 below was adapted from Phong's original paper to ensure physical plausibility:

$$\begin{aligned} & f_r(x, \Psi_i \rightarrow \Psi_o) \\ & = \frac{\rho_d}{\pi} + \frac{\rho_s \left(\frac{s+2}{2\pi} \cos^s \phi\right)}{|\cos(N_x, \Psi_i)|} \end{aligned} \tag{2.7}$$

As in equation 2.5, ϕ is the angle between the specular scatter direction R and the outgoing radiance direction Ψ_o.

The Phong BRDF's cosine lobe is symmetric around R. Blinn [12] as well as Cook and Torrance [13] on the other hand first model the surface as being composed of a distribution of specular microfacets and then the BRDFs resulting from their microfacet models. The angle between the average surface normal and a microfacet normal is related to the half-angle θ_H, but the PDF of a vector scattered from the microfacets is not symmetric around R. A sketch by Ngan et al. [14] highlights the difference in beam shape between the two formulations of the specular lobe.

Part III discusses the important properties required of BRDFs to be physically plausible. Physically plausible formulations of some of the well known BRDFs are also given.

2.3 The Rendering Equation

The physics of geometric light transport has been brought together in Kajiya's [2] rendering equation. The notation and formulation of the rendering

equation for reflection used here is as described by Dutré et al. [3]:

$$L(x \to \Theta) = L_e(x \to \Theta) + \iint_{\Omega_x} f_r(x, \Psi \to \Theta) L(x \leftarrow \Psi) |\cos(N_x, \Psi)| d\omega_\Psi$$

The rendering equation states that the radiance leaving point x in direction Θ, $L(x \to \Theta)$, is equal to the sum of the radiance emitted and the radiance reflected by the surface. The reflected radiance is due to the irradiance collected over the hemisphere Ω_x above x. The double integral over the hemisphere is an application of Equation 2.3 to calculate the total scattered radiance in direction Θ.

The algorithms that have been designed to approximately solve the rendering equation are known as *rendering algorithms*. One distinction between the algorithms is the *direction* of rendering. The three main types of rendering algorithms are then:

- *Forward rendering*, solving the rendering equation using transport paths from the camera.
- *Backward rendering*, solving the rendering equation using transport paths from the light source.
- *Bi-directional rendering*, partially solving the rendering equation from both the light source and camera then bringing the solutions together somewhere in between.

Often a single rendering algorithm is able to only partially solve the rendering equation. Typically such a partial solution of the rendering algorithm results in the simulation of only a subset of the light transport path types.

To solve the rendering equation Kajiya himself described a Monte Carlo (MC) *forward rendering* algorithm known as path tracing. Kajiya's path tracing may be used to effectively solve $LD(D|G|S)^+E$ transport paths[4].

[4]Heckbert's [1] regular expression notation for light transport paths is useful for clearly describing the transport paths solved by the different rendering algorithms.

However, caustic $L(S|G)^+D(D|G|S)^+E$ paths for example are near impossible to render with forward rendering algorithms such as path tracing.

Note that a *single-bounce* caustic is due to transport paths that start at the light and encounter *one* specular or glossy surface interaction before encountering a diffuse receiving surface. Such a path may be written as $L(S|G)^1D$. A *second-bounce* caustic is due to transport paths that have encountered *two* specular or glossy surface viz. $L(S|G)^2D$. A multi-bounce caustic is due to transport paths including one or more specular or glossy interactions viz. $L(S|G)^+D$.

Among the rendering algorithms that Dutré et al. [15] discuss, the MC backward rendering light tracing (LT) algorithm is useful as a reference renderer for caustic $L(S|G)^+D$ transport paths. It is indeed used later in the thesis as the reference renderer for specular and glossy caustics. In light tracing, many photons each representing a quantum of energy are emitted from the light sources. The photons' geometric light paths are then traced through the scene. The choices of emission directions, scatter incidents and the scatter directions are chosen randomly based on a number of PDFs. In the light tracing algorithm, each time a photon is scattered *or* absorbed along its path, a ray is also cast to the camera's aperture to provide a weighted steady state contribution of radiance to the camera film. The light tracing rendering algorithm, although suitable for rendering caustics, is not effective at rendering the camera image of specular or very smooth glossy objects. The reason for this becomes evident if one calculates the radiance reflected from a surface using the definition of the BRDF in Equation 2.2:

$$dL(x \rightarrow \Psi_o) \quad = \quad f_r(x, \Psi_i \rightarrow \Psi_o)L(x \leftarrow \Psi_i)|\cos(N_x, \Psi_i)|d\omega_{\Psi_i}$$

The sharper the BRDF peak the more photons—and therefore more Ψ_i direction samples—would have to be sent from the light source to adequately reduce the error. Adequately reducing the error results in a *converged solution.*

A well known *bi-directional rendering* algorithm is bi-directional path tracing [16][9] which combines path tracing with light tracing. Light trans-

port paths are constructed by concatenating random sub paths from the camera with random sub paths from the light sources. Bi-directional path tracing is more general than path tracing and may be used to effectively solve most of the $L(D|G|S)^+E$ transport paths. Some other bi-directional rendering algorithms such as photon mapping (PM) first cache the light sub-paths (or typically a representative subset thereof) for use during a second forward rendering phase. Using a cached subset of the light paths causes the variance of the solution to reduce quicker, but at the risk of increasing the bias in the solution. The next chapter, Chapter 3, discusses more rendering algorithms that are useful for rendering caustics.

2.4 Spherical Gaussian Functions

A spherical Gaussian (SG) is a type of spherical radial basis function (SRBF) which will be used later in the thesis. The SG definition used in this thesis is:

$$G(\Theta) = \frac{1}{2\pi\sigma^2} e^{-\frac{\phi^2}{2\sigma^2}} \tag{2.8}$$

$$\approx \frac{1}{2\pi\sigma^2} e^{-\frac{\|\Theta - R\|^2}{2\sigma^2}} \tag{2.9}$$

$$\text{where} \quad \vec{\phi} = \Theta - R$$

$$\text{and} \quad \phi \approx \|\vec{\phi}\|$$

In equation 2.8, ϕ is the displacement angle from some reference vector $\vec{R}$ to the function vector parameter $\vec{\Theta}$ and σ is the standard deviation of the PDF. The normalisation $\frac{1}{2\pi\sigma^2}$ used for G is the 2D Gaussian normalisation, but it is valid on the sphere for the relatively small ($< \frac{\pi}{4}$) standard deviations used for glossy surface interactions. Equation 2.9 is a small angle approximation sometimes used when $\phi < \frac{\pi}{6}$. An important property of SGs is that the convolution of two SGs is a third SG with variance equal to the sum of the variances of the SGs.

The SG definition used here is different from the definitions used elsewhere in the literature which contain a $e^{\lambda(\Theta \cdot R - 1)}$ and effectively take e to the

power of $\cos\phi - 1$ as opposed to $-\phi^2$. In such cases λ represents the lobe sharpness instead of a standard deviation as σ does. The advantage of using the standard deviation is that it has intuitive meaning with regard to the scatter lobe.

Another advantage of the SG definition used in this thesis is its similarity to a 2D Gaussian when working in a Cartesian (u, v) plane viz. $\vec{\phi} = a\hat{u} + b\hat{v}$. Such a domain is related to the small angle approximation shown in Equation 2.9 and fits in well later in the thesis.

2.5 The Problem and Thesis Scope

One of the challenges for any rendering algorithm that traces individual rays of light is the number of computations that is required to simulate enough transport paths to adequately reduce the error in the render result. LBT is a two phase bi-directional algorithm which is especially efficient for rendering caustics. The algorithm is able to exploit the coherency of the transport paths within an envelope or beam of light. In other words, the algorithm minimises the number of light sub-paths required for convergence of the result. LBT rendering algorithms found in the literature unfortunately only support mirror-like specular surface interactions.

The scope of the work encompasses both the derivation of the new physically based light beam rendering algorithm that supports glossy surface interactions as well as a multi-core CPU implementation of it and of other reference rendering algorithms. If the number of overlapping beams illuminating a surface fragment is controlled then, even on the CPU, the algorithm becomes efficient enough for interactive image synthesis. Nonetheless, the main focus of the thesis is on the derivation of a new light beam radiance estimate[5]. Future work could focus on interactive or real-time implementation using, for example, many-core CPUs and/or the Graphical Processing Unit (GPU).

The study into LBT and the proposed light beam radiance estimate

[5]The algorithm to estimate the radiance is often referred to simply as the radiance estimate.

should in no way conclude with this thesis. On the contrary, Part VI critically reflects on the success of the undertaking, highlights limitations and then points ahead to future work. The software that was written for the thesis and used to generate the results is available at: `http://code.google.com/p/stitch-engine/source` tag sprinkles[6].

[6]The motivation for the tag name is given in `CoffeeShake.txt` in the code repository.

Chapter 3

Related Work

This chapter gives a brief overview of the related work to the topic of LBT. The early development of rendering is presented followed by the related work on firstly beam tracing for rendering caustics and then the use of spherical Gaussians in rendering.

3.1 Early Work

The recursive ray trace rendering algorithm published in 1980 by Whitted [17] is well known and still widely used. Whitted was not the first to use ray tracing for rendering, but his algorithm, known as Whitted ray tracing, is simple and produces realistic specular reflections as shown in Figure 3.1.

In 1984, after Whitted [17] had shown how ray tracing from the camera could be used for realistic shaded rendering (LDS^*E transport paths), Heckbert and Hanrahan [18] attempted to improve upon the rendering performance by tracing polygon beams from the camera. The main difficulty was in clipping the beams against the scene geometry. Arvo [4] showed in 1986 that high frequency lighting effects such as caustics—which was difficult with the approaches of Whitted, and of Heckbert and Hanrahan—could be rendered by backward ray tracing *from the light* and using illumination maps. However, the scene geometry had to be discretised into polygons and to obtain good results a lot of storage and many light rays were required

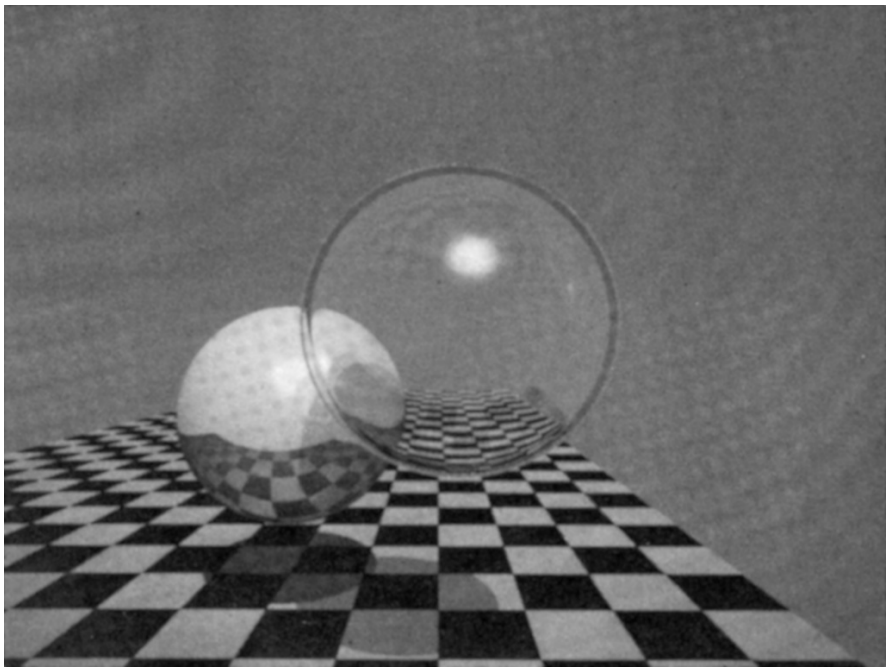


Figure 3.1: Classic example of Whitted ray tracing taken from taken from [17].

to generate high resolution illumination maps. An example of backward ray tracing is shown in Figure 3.2.

Around this same time the physics of geometric light transport was brought together by Kajiya [2] in the rendering equation. A rendering algorithm is an algorithm that has been designed to solve the integral rendering equation and Kajiya describes the Monte Carlo (MC) path tracing rendering algorithm.

Nishita et al. [19] used non-uniform illumination volumes and the scene's polyhedron contours to project shadow volumes. A scan-line algorithm was used to render single scattering of participating media. Shinya et al. [20] combined pencil tracing[1] and backward ray tracing into what they called grid-pencil tracing. Because they scan-line converted the resulting illumination polygons into an *illumination map*, their approach unfortunately suffered from similar limitations than Arvo's backward ray tracing.

In 1995, Arvo [21] published a paper on using *irradiance tensors* for,

[1]Pencil tracing traces a number of paraxial rays. The paraxial bundle is more efficient to trace than so many individual rays.

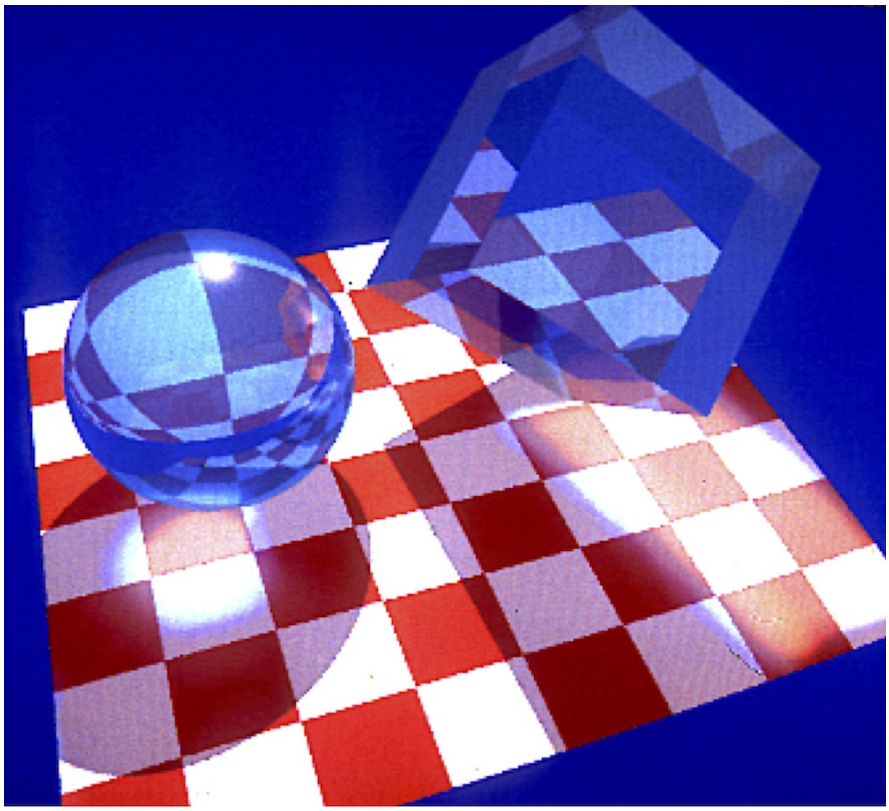


Figure 3.2: An example of Arvo's backward ray tracing taken from taken from [4].

amongst other things, calculating the irradiance due to directional area light sources and the appearance of glossy surfaces. Arvo's rendering algorithm follows from Lambert's formula for calculating irradiance. A drawback of Arvo's algorithm is that the performance of calculating the irradiance due to a single beam is dependent on the distribution (e.g. specular, narrow or wide) of the scattered or emitted radiance. Although Arvo does not show or discuss rendering of caustics it could, nevertheless, be quite enlightening to study the potential duality between irradiance tensors and the proposed LBT approach.

3.2 Light Beam Tracing and Caustics

The *LSDE* backward (read 'light') beam tracing approach of Watt [5], published in 1990, alleviated the aliasing and illumination map memory requirements of Arvo's backward ray tracing. This was done by tracing polygon beams from the light as opposed to individual rays. Each specular polygon in the scene scattered the light energy as a polygon beam which was then projected onto diffuse surfaces and used as lighting polygons. Shadowing was handled by subdividing a partially lit specular polygon and optionally also by firing shadow rays back towards the scatterer polygon. Watt's approach however did not support multi-bounce transport paths and only considered mirror-like specular surfaces. An example of BBT is shown in Figure 3.3.

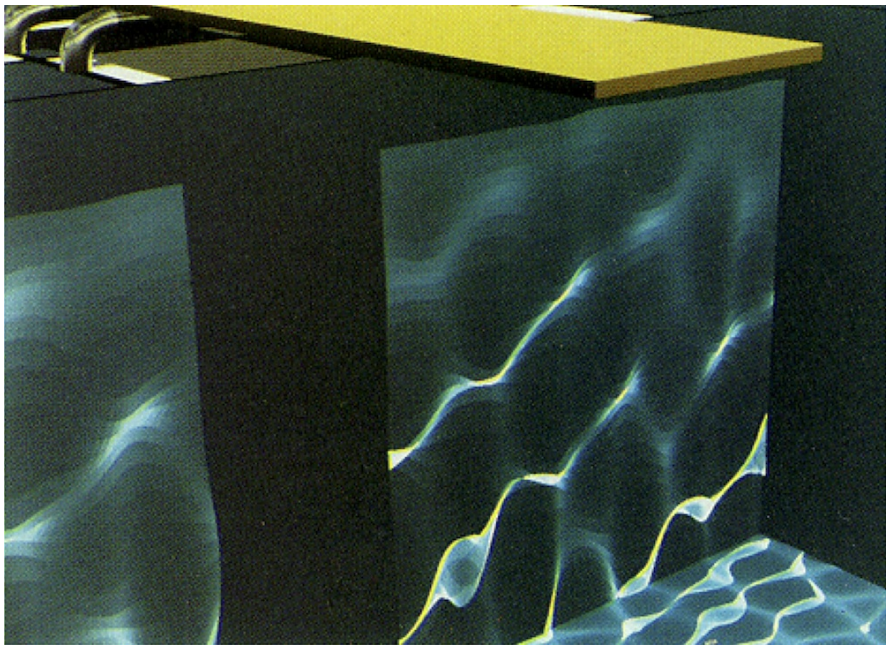


Figure 3.3: An example of Watt's backward beam tracing (taken from [5]) using approximately 10k beams.

Mitchell and Hanrahan [22] used implicit surfaces and numerical techniques to find the illumination due to multi-bounce specular interactions. Nishita and Nakamae [23] use illumination volumes and a scan-line algorithm to directly render the shafts of light, the caustics and simple shadows.

In 1995, Jensen and Christensen [24] replaced Arvo's illumination map with the kd-tree data structure called the photon map and a kernel density estimate based algorithm for estimating the surface radiance due the photons. The algorithm to estimate the radiance is often referred to simply as the radiance estimate. The most important benefit of using a photon map was the decoupling of the lighting information from the scene geometry which solved the memory requirement problem of Arvo's backward ray tracing. However, for the radiance estimate to converge one still potentially required many backward rays or photons to be traced. An example of photon mapping (PM) used for caustics is shown in Figure 3.4.

Figure 3.4: An example of Jensen's photon mapping used for caustics taken from [25]. The caustic is rendered using approximately 350k photons.

Also in 1995, Chuang and Cheng [26] improved on Watt's [5] BBT. They used a beam bounding volume hierarchy (BVH) and a sub-linear performance point-in-beam test to efficiently find all beams that contribute illumination to a receiving surface point. Their approach also allowed efficient illumination of curved and irregular surfaces.

Brière and Poulin [27] further improved on the BBT algorithms proposed

by Watt and by Chuang and Cheng in [5, 26]. Their improvement, published in 2001, relied on a *light image* to setup the beam paths. Similar to the photon map, the light image decoupled the lighting from the scene geometry. The light image also simplified the light transport enough that multi-bounce specular transport paths could be implemented. Brière and Poulin made use of a bounding volume hierarchy (BVH) to accelerate point-in-beam detection. An example of the light image and BBT (or LBT) is shown in Figure 3.5.

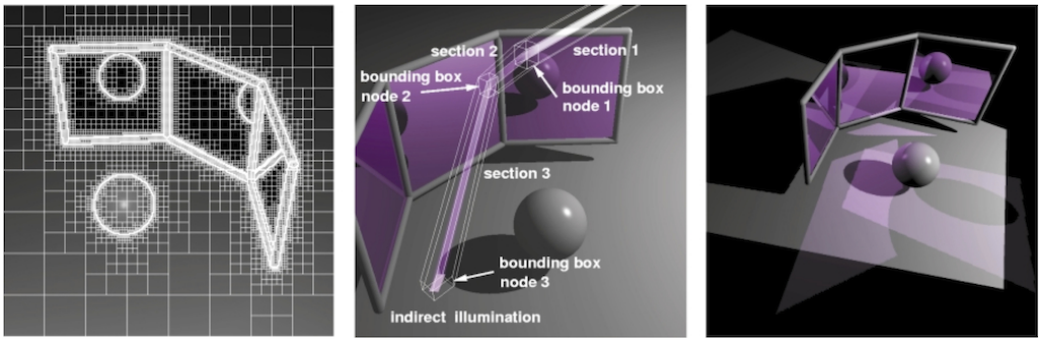


Figure 3.5: An example of Brière and Poulin's light image used for caustics taken from [27]. Approximately 2.5k beams were used.

Iwasaki et al. [28] and Iwasaki et al. [29] made use of the GPU to speed up Nishita and Nakamae's [23] scan-line illumination volume technique. Ernst et al. [30] also discussed a GPU accelerated BBT implementation.

In 2007, Schjøth et al. [31] improved upon PM by adding the differential wavefront information by using photon differentials. Photon differentials are based on ray differentials by Igehy [32] and Suykens et al. [33]. The photon differentials allowed one to propagate the photon footprints and dynamically adapt the size and shape of the filter kernel used in the radiance estimate. This minimises the bias in the solution while maintaining low variance of caustics and shadows. However, the support of the kernels could be incomplete in low photon density areas, leading to noise.

In 2008, Jarosz et al. [34] improved on PM to include a beam radiance estimate for efficient rendering of participating media. However, they still had to trace and store individual photons, which was expensive. Luckily, progressive rendering techniques [35][36] brought the advantage of delivering

an approximate or best possible solution within a specified resource budget such as a time or memory limit.

Spencer and Jones [37] proposed, in 2009, to add a photon relaxation step to PM that redistributes the photons before the forward render step and photon density estimates. Photon relaxation permits a smaller kernel to be used in the radiance estimate which reduces the bias, but due to the redistribution a low variance is maintained. Spencer and Jones [38] further improved upon photon relaxation by parameterising the photons to distinguish between neighbouring light ray envelopes. They also developed a progressive solution [39] to photon relaxation. However, the drawback of photon mapping remains: many individual photons have to be traced from the light source, although not as many as for classical PM.

In 2010, Chen et al. [40] combined standard PM with the light beam concept by grouping photons with similar transport paths into light beams with polygonal boundaries. They make use of a photon radiance estimate which requires many photons to converge. However, their ideas on how to isolate light envelopes using particle tracing could potentially be incorporated into the LBT algorithm developed in this thesis.

Specular beam tracing has been useful in rendering applications where efficiency is important. In the last ten years, rendering algorithms similar to that of Briere and Poulin [41] have been used by Ernst et al. [30], Hu et al. [42], Liktor et al. [43] and others for real-time rendering applications.

In 2011, Jarosz et al. [44] discussed a very interesting photon beams rendering algorithm which improved upon the efficiency of the beam radiance estimate of Jarosz et al. [34] and others. Using photon beams one does not need to generate and store as many photon scatter incidents because the radiance estimate is done on the whole photon path directly. The rendering algorithm did, however, still require individual photon beam paths to be traced. A progressive rendering solution [45] was used in practice, because tracing many photon beam paths is quite expensive in terms of computation and memory. It will be shown in subsequent chapters that LBT mitigates this expense.

In 2012, Georgiev et al. [46] wrote a paper on the vertex connection and

merging of light paths which recasts PM as a path sampling technique that may be used to improve Monte-Carlo rendering algorithms such as bidirectional path tracing. In 2014, Křivánek et al. [47] again used the path integral formulation of light transport to also unify rendering algorithms such as photon beams, discussed above, with, for example, bidirectional path tracing.

3.3 Spherical Gaussians and Caustics

The use of Gaussian SRBFs (or SGs) in rendering have become quite popular. In 1992, Ward [48] presented isotropic and anisotropic Gaussian BRDFs. In 1996, Tsai and Shih [49] used these functions to accelerate the precomputed radiance transfer (PRT) rendering algorithm for glossy objects. In 2009, Wang et al. [50] used SGs for real-time rendering of dynamic, spatially-varying BRDFs in static scenes. In 2012, Iwasaki et al. [51] introduced the Integral Spherical Gaussian (ISG) used to efficiently evaluate the integral of a SGs over an axis-aligned rectangle in spherical coordinates. ISG is related to summed-area table (SAT). They used ISG for efficient rendering of dynamic scenes incorporating all-frequency BRDFs. All-frequency BRDFs include all BRDFs from specular to glossy to diffuse.

Spherical Gaussians have not been used for rendering caustics with the exception of Xu et al.'s [52] work. In 2014, Xu et al. [52] used spherical Gaussians to render single bounce *all-frequency* interactions. They used a spherical Gaussian representation of the light source and of the BRDF to simplify the rendering equation to a single decomposable surface integral to which they apply a line integral approximation. The authors built a tree of scene primitives and used a tree cut (reflector cut) to arrive at an approximate render solution within an error, a time or another resource budget. However, unlike light beams, Xu et al.'s approach does not support multiple bounces. Moreover, the binary tree of scene primitives limits the applicability of the approach to tessellated scenes.

3.4 Summary

As mentioned for the irradiance tensor work of Arvo [21], it should be enlightening to investigate the duality between various rendering algorithms including Xu et al.'s work, Arvo's irradiance tensors and my work. However, for the purposes of this thesis, I persist in developing the spherical Gaussian LBT rendering algorithm which I undertook in 2009. The further investigations—including recasting LBT in the path integral formulation of light transport—is left as future work.

Part II

The Glossy Light Beam Tracing Conjecture

This part of the thesis puts forward a conjecture on an irradiance estimate that is expected to lead to a LBT rendering algorithm that supports glossy surfaces. Most of these ideas were originally published in research papers [53][54].

Chapter 4

Classical Backward Beam Tracing

This chapter gives an overview of the classical BBT rendering algorithm developed by Watt and afterwards improved by Chuang and Cheng [26], and Brière and Poulin [27]. The next chapter (Chapter 5) will then offer a conjecture on extending LBT to include glossy interactions.

As mentioned in the introduction, the term *backward tracing* is used interchangeably with light tracing to describe ray or beam tracing *from the light* in the same direction as the flow of radiance. Forward tracing and forward rendering is used to describe ray or beam tracing from the camera in the same direction as importance[1] [3]. This convention ties up with the convention of classical BBT [5].

4.1 Overview

Like classical PM there are two phases to rendering an image using light beams: a light phase and a gather (forward rendering) phase. The light phase constructs the LS^*D transport paths while the forward render phase connects these transport paths to the eye which, in the case of using a Whitted forward renderer, result in LS^*DS^*E transport paths.

[1] Importance is the adjoint of radiance as an importon is the adjoint of a photon.

Light beam tracing (LBT) proceeds as outlined in Algorithm 1. The `Render` procedure shows the call to `TraceBeams`, followed by a parallel for-loop over the pixels in the image to gather the radiance for each pixel in the image. The light phase is implemented in the `TraceBeams` procedure and the forward phase is implemented in lines three to five using the `Gather` procedure.

In LBT, the aim is to trace light beams instead of individual rays from the light source into the scene. A beam traced past its first bounce is shown in the left subfigure of Figure 1.2. Tracing beams reduces the number of transport operations needed to render a scene, and this results in more efficient rendering.

The outer boundaries of the beam are referred to as the beam's corner rays. Corners are shown as yellow arrows in the left subfigure of Figure 1.2. Tracing a light beam is accomplished by tracing the corner rays of the beam. As mentioned before, LBT is efficient because the algorithm exploits the coherency of the transport paths within a light envelope or beam. In other words the transport paths within the beam are similar to the corner paths.

A specular beam describes a light beam scattered by a mirror-like specular surface. A glossy beam describes a light beam scattered from a glossy surface.

Chuang and Cheng's improvement of BBT is chosen as a starting point as opposed to the more complex light image based approach of Brière and Poulin. This offers a simpler starting point for rapidly testing the conjecture presented in the next chapter. The more complex light image based approach is subsequently used, however, to implement single- and multi-bounce glossy LBT, as discussed in Part IV of the thesis.

4.2 Tracing Light Beams

This section presents the details surrounding the `TraceBeams` procedure. Each scene primitive polygon scatters a light beam from every light. For each polygon, light rays are traced from each light to its vertices and scattered (reflected or refracted) as the corner rays of the scattered beams. Only single scattering of beams is initially considered. Occlusions and multiple

Algorithm 1 Light beam tracing.

```
1: procedure RENDER                                   ▷ Renders one image frame.
2:     TRACEBEAMS
3:     for all pixel ∈ image do                                  ▷ in parallel
4:         L_pixel = GATHER(CameraRay(pixel))
5:     end for
6: end procedure

7: procedure TRACEBEAMS
8:     ...                                            ▷ Trace light beams.
9:     ...                                              ▷ See Section 4.2.
10: end procedure

11: procedure GATHER(ray)
12:     intersect = scene.calcIntersection(ray)

13:     bList=CBVH.getContributingBeamSegments(intersect)

14:     for all b ∈ bList do
15:         Area_⊥ = b.calcOrthArea(intersect)
16:         Ψ̄ = b.calcDirection(intersect)
17:         N_i = intersect.normal
18:         Φ = b.flux = ρ_s Φ_s
19:
20:         ray.L += (ρ_d/π) (|Ψ̄·N_i| / Area_⊥) × Φ          ▷ See Equation 4.2
21:     end for

22:     ray.L += intersect.L_e

23:     GATHER(intersect.specReflRay)          ▷ Whitted forward render...
24:     ray.L += ρ_s · intersect.specReflRay.L
25: end procedure
```

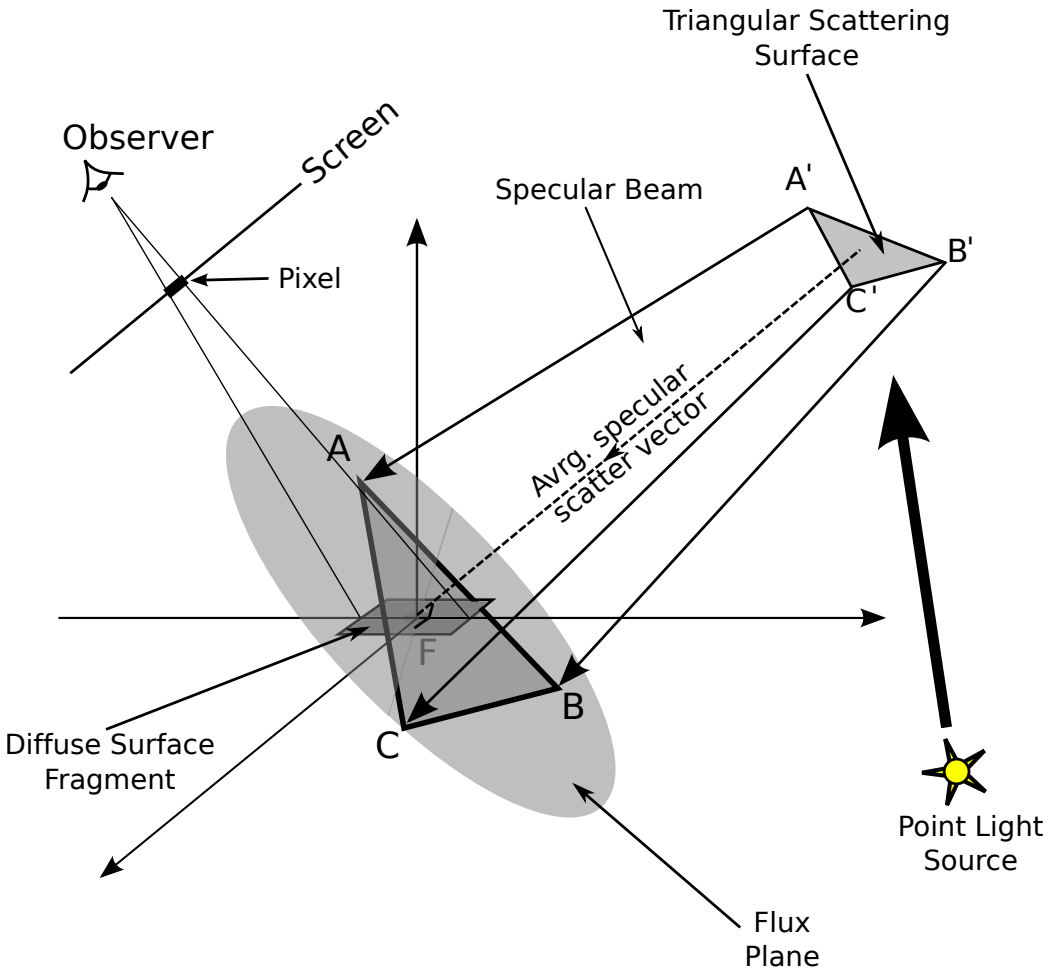


Figure 4.1: The orthogonal flux plane defined by triangle ABC.

bounces are ignored.

Each scattered light beam is modelled by individually scattered light vectors from each of the polygon vertices (A', B' and C') as shown for a reflected beam in Figure 4.1. Each beam transports an amount of flux which is conserved within the beam. An advantage of doing it this way is that the smooth surface (as represented by the scattering surface's mesh normals) and the scattered light beams are conveniently decoupled from the polygon's geometry.

During forward rendering, an estimate must be made of the irradiance incident on the surface fragment behind every pixel. A beam bounding volume

hierarchy may be used to efficiently find all beams that include the surface fragment and adds to the surface irradiance. Such a bounding volume hierarchy is often referred to as a task acceleration structure. The task being to find all beams that include the query point.

For the proof of concept LBT rendering algorithm a cone bounding volume hierarchy (abbreviated to CBVH and described in more detail by Chuang and Cheng [26]) is built from all the scattered beams. The call `CBVH.getContributingBeamSegments(intersect)` executes the CBVH query for the ray surface intersection and returns the list of beams that contribute irradiance to that point.

A Whitted style ray tracer was chosen for the forward rendering. However, if an OpenGL or other hardware accelerated forward renderer is used instead of the ray trace forward renderer discussed in the thesis then a screen-space bounding volume approach with a highly optimised inner render loop might be used as opposed to a CBVH. Such an approach is proposed by Ernst et al. [30]. The first concept demonstrator for glossy LBT described in one of my earlier papers [53] used an even more brute force linear list of beams and still achieved real-time rendering of glossy beams and caustics on graphics hardware.

4.3 The Irradiance Estimate

This section presents the details surrounding the `Gather` procedure. Once the list of beams that contain the receiving point x is retrieved from the CBVH, then the radiance due to each beam may be calculated.

The beam flux, Φ_s, from the light is conserved within the beam. The subscript s is in reference to the fact that the flux is reflected from a mirror-like specular $s$urface. The flux is calculated from the radiant intensity of the light source and the solid angle subtended by the beam from the point light. If the distance to the light is relatively large compared to cross sectional area of the beam then the orthogonal flux density at x may be approximated by

$\frac{\Phi_s}{A_{x\perp}}$ and the irradiance at x is then given by:

$$E(x) \approx \rho_s \frac{\Phi_s}{A_{x\perp}} |\cos\left(N_x, \overline{\Psi}\right)| \tag{4.1}$$

N_x is the surface normal at x and $\overline{\Psi}$ is the *average* or approximate direction of the specular light path incident at x. $\overline{\Psi}$ is a good representation of the incident light directions as long as the light source is relatively far away. $A_{x\perp}$ is the cross sectional area of the beam at x and ρ_s is the specular reflectivity.

The view independent diffuse reflected radiance L for each beam may be calculated from the definition of the diffuse BRDF given in Equation 2.4

$$L(x \to \Theta) = \frac{\rho_d}{\pi} E(x) \tag{4.2}$$

where ρ_d is the diffuse reflectivity. The radiance is independent of the direction vector Θ from x to the *eye*, but Θ is included for context.

The virtual flux plane, defined by ABC in Figure 4.1, is used to calculate $A_{x\perp}$ for every x. The flux density is therefore accurately calculated even for receiving surfaces that are not orthogonal to the beam. Unlike Ernst et al. [30] I do not explicitly average the flux density over neighbouring beams.

To lead into the next section on the glossy LBT conjecture the expression of the beam irradiance in equation 4.1 is updated slightly. Using a Dirac Delta as an on/off switch (triggered by whether or not x is within the beam) I would like simply to express the irradiance due to a beam reflected from a mirror-like *specular* surface as:

$$E(x) = \frac{\Phi_s}{A_{x\perp}} |\cos\left(N_x, \overline{\Psi}\right)| \iint_{\Omega} \delta(\vec{\phi}) d\omega_{\phi}. \tag{4.3}$$

Figure 4.2 attempts to better explain this by showing a 2D side-view of a single bounce beam scattered by a smooth specular surface. Note that transmissive scattering is shown for simplicity. The Dirac delta function is drawn as a sharp peak. Ω is not a solid angle domain of integration. It is the domain of all $\vec{\phi}$ vectors that represent the reflecting surface in the 2D Euclidean domain of the Dirac delta function. The over arrow indicates a

vector quantity.

The irradiance integral proceeds from ϕ_1 at B' through zero to ϕ_0 at A'. The size of the integration domain is then:

$$\Omega_{1D} = \phi_1 + \phi_0. \tag{4.4}$$

The leap from Equation 4.1 to Equation 4.3 is a considerable one, but ultimately shown to be correct in the next part, Part IV, of the thesis. For now it is sufficient to note that:

- The Dirac delta has the desired effect of acting as an on-off switch for the beam irradiance depending on whether or not x is within the beam.
- The on-off switch is *all* that has been added from Equation 4.1 to Equation 4.3.

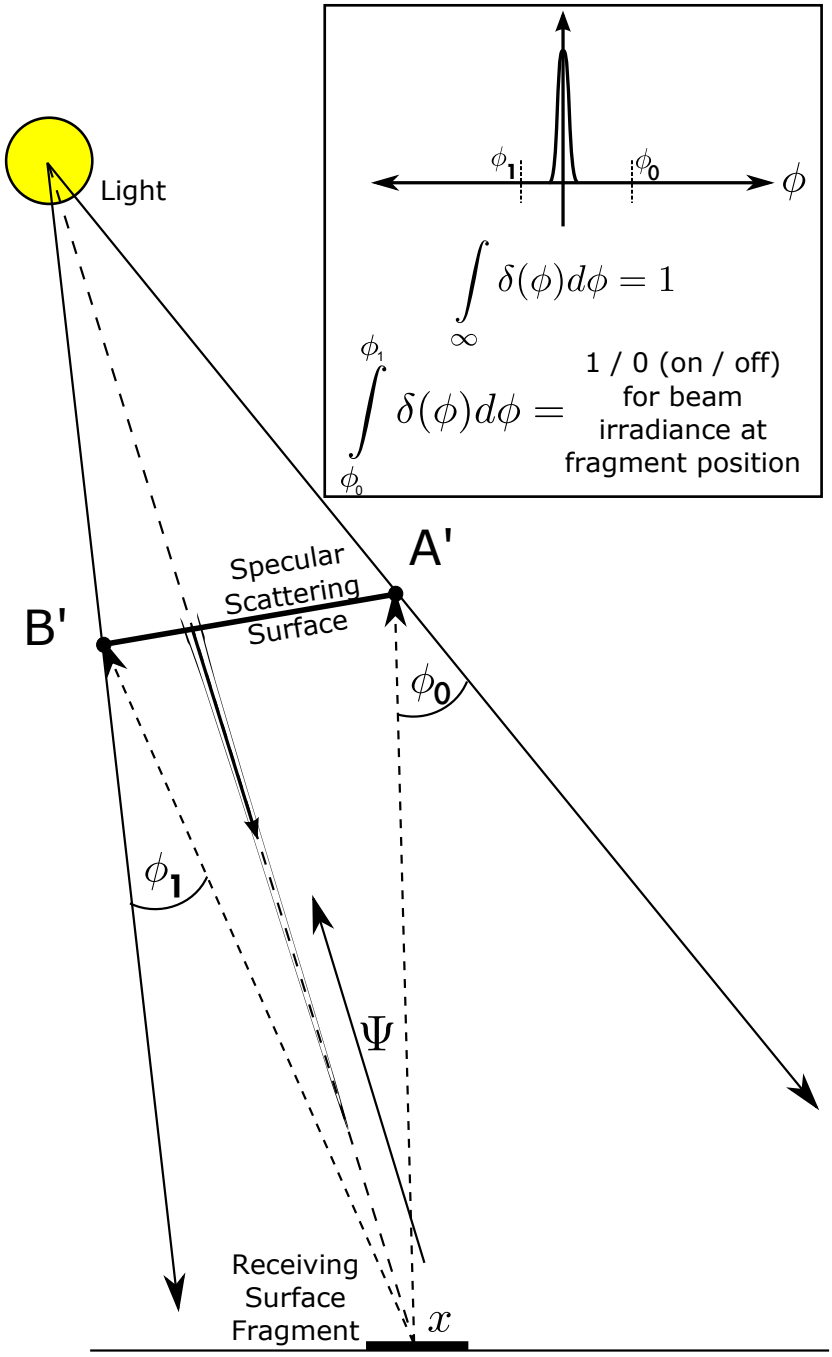


Figure 4.2: A 2D view of a single bounce light beam scattered by a smooth specular surface. Note that transmissive scattering is shown for simplicity.

Chapter 5

Conjecture

This chapter offers a conjecture on an irradiance estimate that is expected to lead to a LBT rendering algorithm that supports glossy surfaces. Classical single bounce BBT is extended to include glossy surface interactions. The next chapter then tests and adds weight to the conjecture by showing some render results.

Here, an overview of the reasoning behind the proposed approach for glossy LBT is given. This is followed by the proposed glossy irradiance estimate.

5.1 Overview

The classical beam irradiance equation, Equation 4.3, from the previous chapter is repeated below for easy reference:

$$E\left(x\right) = \frac{\Phi_s}{A_\perp} |\cos\left(N_x, \overline{\Psi}\right)| \iint\limits_{\Omega_{ABC}} \delta(\vec{\phi}) d\omega_\phi. \tag{4.3}$$

The Dirac delta function is a PDF which describes the distribution of the scattered vectors of light around the specular reflection direction. The Dirac delta is zero everywhere except along the specular scatter direction implying that light is only scattered along the mirror-like specular scatter direction.

This raises two diverting questions:

- Can one use other PDFs in the place of the Dirac delta PDF to model reflected beam irradiance in cases where the reflecting material is not a smooth specular surface?
- Can one derive a physically plausible BRDF that results in such a PDF of scattered light?

The first question is explored further in the rest of this chapter while the second is tackled in Part III of the thesis.

5.2 The Glossy Irradiance Estimate

The basic conjecture is, therefore, that in order to include glossy surface interactions in LBT, the Dirac delta PDF of Equation 4.3 should be replaced with another scatter PDF that has larger support.

I propose that a SG PDF be used, denoted by ρ. The general shape of such a PDF is shown on the right in Figure 5.1. As discussed in Section 3.3 of the related works chapter, SG PDFs have already been used by other authors for representing BRDFs and recently for rendering caustics. SGs are more efficient to evaluate than, for example, a cosine PDF and the convolution of two SGs is a third SG.

However, the SG formulation $\rho(\phi) = \frac{1}{2\pi\sigma^2} e^{-\frac{\phi^2}{2\sigma^2}}$ (discussed in Chapter 2) is different than used by other authors. The SG definition used here is similar to a 2D Gaussian when working in a Cartesian (u, v) plane viz. $\vec{\phi} = a\hat{u} + b\hat{v}$ which becomes useful later in the thesis.

When replacing the Dirac delta PDF with a SG PDF the irradiance at x due to a light beam becomes:

$$E(x) = \frac{\Phi_s}{A_\perp} |\cos\left(N_x, \overline{\Psi}\right)| \iint_{\Omega_{ABC}} \rho(\vec{\phi}) d\omega_\phi. \tag{5.1}$$

It is again worth noting that:

- The Gaussian PDF has the desired effect of spreading the beam flux to outside of the specular beam.

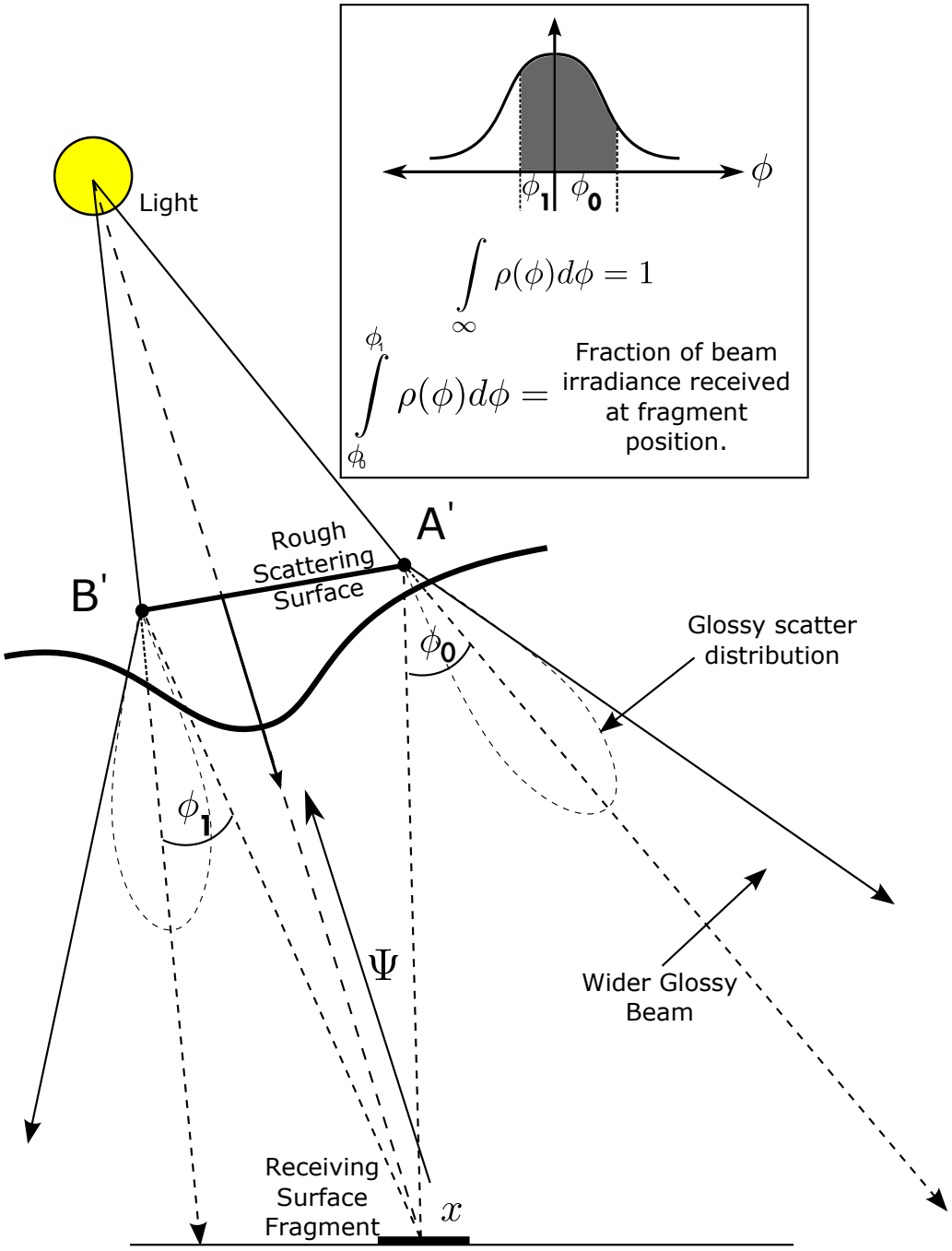


Figure 5.1: A 2D view of a single bounce light beam scattered by a glossy surface. Note that transmissive scattering is shown for simplicity.

- The Dirac delta PDF of Equation 4.3 is swapped for another PDF which does not affect the total flux transported by a narrow beam.

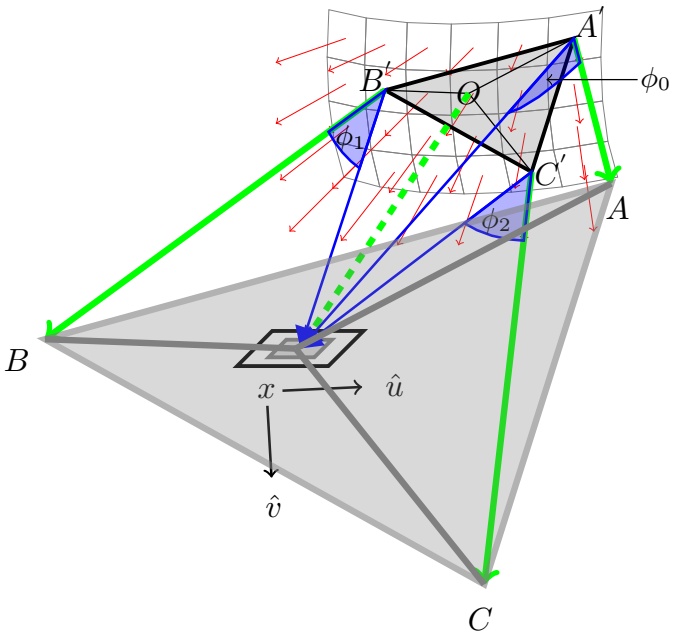


Figure 5.2: The beam's corner light paths are shown in green. A, B and C define the orthogonal flux plane through x on the receiving surface.

Ω_{ABC} is the 2D equivalent of the 1D domain Ω_{1D} referred to earlier in Equation 4.4 and now shown in Figure 5.1. Ω_{ABC} is the domain of $\vec{\phi}$ vectors for all points within triangle $A'B'C'$ in Figure 5.2.

Figure 5.3 shows a different view of a single bounce light beam. $E(x)$ is the irradiance incident at x due to the light beam reflected from the glossy surface at y. The surface at x is diffuse and therefore the radiance perceived by the eye is independent of the direction vector Θ from x to the *eye*, but Θ is included for context.

As mentioned, $\vec{\phi}$ is the offset of the outgoing direction relative to the specular scatter direction R from y in Figure 5.3. The outgoing direction is now called $-\Psi$ as opposed to Ψ_o used earlier. R may be calculated using the

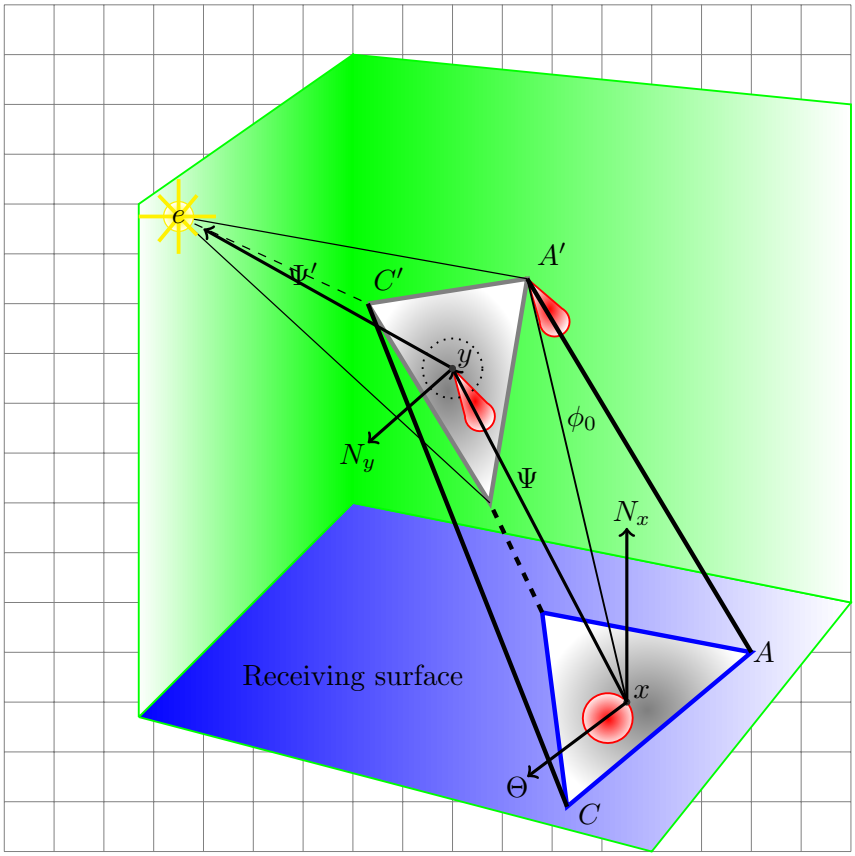


Figure 5.3: The surface at y (a free surface) scatters a light beam onto a diffuse receiving surface at x. The PDF of the glossy scattered vectors is shown in red at y. The PDF of the scattered vectors due to the diffuse receiving surface at x is shown for context.

dot product which results in:

$$\begin{aligned} R &= 2(\Psi' \cdot N_y)N_y - \Psi' \\ \vec{\phi} &= -\Psi - R \\ \phi &= \|\vec{\phi}\|. \end{aligned} \tag{5.2}$$

The domain Ω_{ABC} is defined by $\vec{\phi_0}$, $\vec{\phi_1}$ and $\vec{\phi_2}$ shown in Figure 5.2 and Figure 5.4. In addition to using Equation 5.2, $\vec{\phi_0}$, $\vec{\phi_1}$ and $\vec{\phi_2}$ may also be

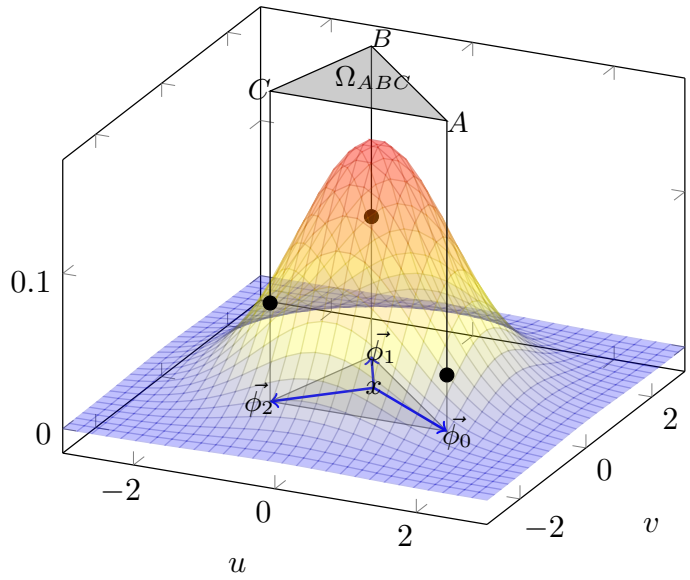


Figure 5.4: The 2D Gaussian scatter distribution is shown—recall the 1D scatter distribution shown earlier in Figure 5.1. The ϕ angles define the triangular domain Ω_{ABC} used in LBT lighting integrals. In this view the specular light path is through x vertically in the centre of the scatter distribution.

expressed in *any* Euclidian ortho-normal basis ($\hat{u}$ and $\hat{v}$) of the orthogonal[1] flux plane using the information in Figure 5.2.

The length $\phi_1 = \|\vec{\phi_1}\|$, for example, is the angle between the specular

[1]The flux plane is orthogonal to the dashed green specular path (shown in Figure 5.2) that includes x on the receiving surface.

light path at $B^{'}$ and the vector from $B^{'}$ to x in Figure 5.2. This leads to:

$$\phi_0 = \arccos\left(\frac{(A-A')\cdot(x-A')}{\|A-A'\|\|x-A'\|}\right) \tag{5.3}$$

$$\phi_1 = \arccos\left(\frac{(B-B')\cdot(x-B')}{\|B-B'\|\|x-B'\|}\right) \tag{5.4}$$

$$\phi_2 = \arccos\left(\frac{(C-C')\cdot(x-C')}{\|C-C'\|\|x-C'\|}\right) \tag{5.5}$$

$$\vec{\phi_0} = \phi_0\left(\frac{A-x}{\|A-x\|}\cdot\hat{u}, \frac{A-x}{\|A-x\|}\cdot\hat{v}\right) \tag{5.6}$$

$$\vec{\phi_1} = \phi_1\left(\frac{B-x}{\|B-x\|}\cdot\hat{u}, \frac{B-x}{\|B-x\|}\cdot\hat{v}\right) \tag{5.7}$$

$$\vec{\phi_2} = \phi_2\left(\frac{C-x}{\|C-x\|}\cdot\hat{u}, \frac{C-x}{\|C-x\|}\cdot\hat{v}\right). \tag{5.8}$$

Figure 5.4 shows the $\vec{\phi_0}$, $\vec{\phi_1}$ and $\vec{\phi_2}$ vectors in the 2D ortho-normal basis ($\hat{u}$ and $\hat{v}$) of the orthogonal flux plane with the value of the probability distribution for each (u, v). The result of the integral in Equation 5.1, viz. $V(\Omega_{ABC}) = \iint_{\Omega_{ABC}} \rho(\vec{\phi})d\omega_\phi$, is the volume under the probability distribution and over the $\vec{\phi_0}$, $\vec{\phi_1}$ and $\vec{\phi_2}$ triangle in Figure 5.4.

The volume integral may be calculated using any of a myriad of numerical methods. Numerical quadrature is however quite expensive especially considering that during forward rendering the volume calculation needs to happen per light beam per pixel on the display. The simplest solution to the problem of efficiency is to replace the quadrature computation with a lookup table.

To limit the dimensionality of such a lookup table the triangular domain Ω_{ABC} defined by $\vec{\phi_0}$, $\vec{\phi_1}$ and $\vec{\phi_2}$ is decomposed into three wedge domains that each include the origin 0. These domains are Ω_{AB0}, Ω_{BC0} and Ω_{CA0} defined respectively by the pairs $(\vec{\phi_0}, \vec{\phi_1})$, $(\vec{\phi_1}, \vec{\phi_2})$ and $(\vec{\phi_2}, \vec{\phi_0})$ as opposed to a triplet $(\vec{\phi_0}, \vec{\phi_1}, \vec{\phi_2})$. The total probability volume $V(\Omega_{ABC})$ may then be expressed

as:

$$V(\Omega_{ABC}) = V(\Omega_{AB0}) + V(\Omega_{BC0}) + V(\Omega_{CA0}) \tag{5.9}$$

This is a common application of Green's theorem similar to finding the area of a polygon given its vertices.

The decomposed volume $V(AB0)$, for example, is a function of two vectors, $\vec{\phi_0}$ and $\vec{\phi_1}$. The distribution is, however, rotationally symmetric and the volume in fact a function of $\|\vec{\phi_0}\|$, $\|\vec{\phi_1}\|$ and the angle θ_{01} between $\vec{\phi_0}$ and $\vec{\phi_1}$. The decomposed volume may therefore be stored in a 3D lookup table indexed by $(\|\vec{\phi_0}\|, \|\vec{\phi_1}\|, \theta_{01})$:

$$\begin{aligned}
E(x) &= \frac{\Phi_s}{A_\perp} |\cos\left(N_x, \overline{\Psi}\right)| \iint_{\Omega_{ABC}} \rho(\vec{\phi}) d\omega_\phi \\
&= \frac{\Phi_s}{A_\perp} |\cos\left(N_x, \overline{\Psi}\right)| \;\; V(\Omega_{ABC}) \\
&= \frac{\Phi_s}{A_\perp} |\cos\left(N_x, \overline{\Psi}\right)| \left(\|\vec{\nu}(\Omega_{AB0}) + \vec{\nu}(\Omega_{BC0}) + \vec{\nu}(\Omega_{CA0})\|\right) \\
&= \frac{\Phi_s}{A_\perp} |\cos\left(N_x, \overline{\Psi}\right)| \left(\|\vec{\nu}(\vec{\phi_0}, \vec{\phi_1}) + \vec{\nu}(\vec{\phi_1}, \vec{\phi_2}) + \vec{\nu}(\vec{\phi_2}, \vec{\phi_0})\|\right) \\
&\quad \vec{\nu}(\vec{\phi_i}, \vec{\phi_j}) = \boldsymbol{\nu}_{Table}(\phi_i, \phi_j, \theta_{ij}) \frac{\vec{\phi_i} \times \vec{\phi_j}}{\|\vec{\phi_i} \times \vec{\phi_j}\|}
\end{aligned}$$

The right-hand rule is followed for the decomposition of the domain. The signs of the decomposed volumes are chosen to be the same as the signs of the cross products—$(\vec{\phi_0} \times \vec{\phi_1})$, $(\vec{\phi_1} \times \vec{\phi_2})$ and $(\vec{\phi_2} \times \vec{\phi_0})$—of the $\vec{\phi_i}$ vector pairs. This is a common application of Green's theorem similar to finding the area of a polygon given its vertices. In Part IV a related theorem namely Gauss's divergence theorem is used to replace the surface integral with a boundary line integral, removing the need for the lookup table.

To avoid having a volume table for every spherical Gaussian distribution used in the scene, each with its own particular variance, only the normalised $(\sigma = 1.0)$ distribution $\boldsymbol{\nu}'_{Table}(\phi'_i, \phi'_j, \theta_{ij})$ is stored in a lookup table. The normalised $\phi'_i = \frac{\phi_i}{\sigma}$ and $\phi'_j = \frac{\phi_j}{\sigma}$ values are therefore used for the lookup.

The limits and resolution of the lookup table does affect the results. I used a table of $512 \times 512 \times 256$ entries representing a domain of 7.5×7.5 standard deviations and $90°$. Using a lookup table limited to triangles of $90°$ instead of $180°$ makes good use of the lookup table resolution.

Inside the code that implemented the ν'_{Table} lookup table, a $\theta > 90°$ is split into two queries of $\frac{\theta}{2}$ each. The ϕ_{01} vector that splits the query into two halves is calculated using the triangle relationship that an angle bisector divides the opposite side proportional to the two adjacent sides:

$$\vec{\phi_{01}} = \left(\vec{\phi_1} - \vec{\phi_0}\right) \frac{\|\vec{\phi_0}\|}{\|\vec{\phi_0}\| + \|\vec{\phi_1}\|} + \vec{\phi_0}$$

This relationship is proved in the appendix of an excellent book by Posamentier and Lehmann [55] on the mathematics (read 'secrets') of triangles.

Chapter 6

Results and Analysis

This chapter presents the results of the glossy LBT conjecture discussed in the previous chapter. It serves as a proof of concept before the rendering algorithm and the supporting software is developed further.

The performance of the proposed glossy LBT is compared to that of specular LBT. However, the quality of glossy LBT is compared to that of PM because classical specular LBT cannot render transport paths with glossy surface interactions.

I implemented a Whitted [17] forward ray tracer that uses either the set of scattered glossy beams or the caustic photon map when shading a fragment. All performance measurements were done on a Macbook Pro with a 2.26 GHz Intel Core 2 Duo CPU running OSX 10.6.6. OpenMP is used to accelerate the ray tracing over the available two CPU cores and all frames are rendered at a resolution of 640x480 with one sample per pixel. The proof of concept tests were done with a very early version of the StitchEngine software. It was capable of tracing 600000 rays per second and do 8000 k-nearest-neighbours (kNN) queries per second for $k = 100$ on the Core 2 Duo hardware.

For the proof of concept, the scatter distribution's standard deviation σ is specified as a function of a percentage P of $\frac{\pi}{8}$ viz. $\sigma = \frac{\pi}{8} \times \frac{P}{100}$. It is referred to as a scatter distribution of P %.

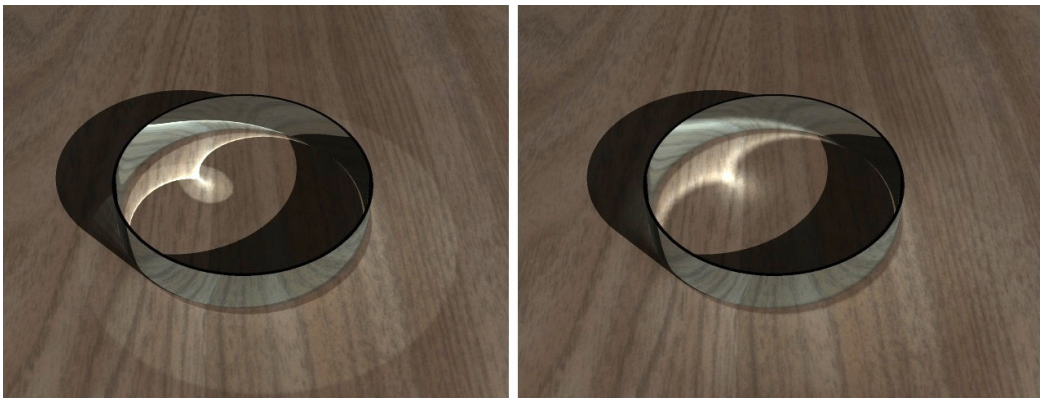


Figure 6.1: A cardioid caustic (75x5 scattering surfaces) rendered with the glossy light beam tracing algorithm. Ring roughness (for caustic transport paths) is increased from left to right.

6.1 Results

Figure 6.2 shows images rendered using glossy LBT. Each image of this simple scene was rendered in approximately 0.16 seconds. Note that a specular beams (requiring a scatter distribution of 0%) is approximated here by a σ of 0.01.

The added cost of the glossy backward beam tracer over that of the specular beam tracer is the cost of the probability density table lookups. The specular backward beam tracer renders an image in 0.15 seconds for the same simple scene. Three table lookups are required for a beam scattered from a triangle primitive, four lookups if scattered from a quad primitive, etc.

Figure 6.3 shows a more complex scene including a diffuse random polytope, a caustic ring, a coloured wall and a refractive water surface with specular (left) and glossy (right) materials. These images were rendered in 4.5 and 7.4 seconds for the 0% and 20% scatter distributions respectively. Note the change in the execution performance of the beam processing for wider beams in the more complex scene. The slower execution is due to the beam overlap.

Figure 6.4 shows a scene with two diffuse Stanford Bunnies, a refractive water surface, a coloured wall and a partial caustic ring. These images were

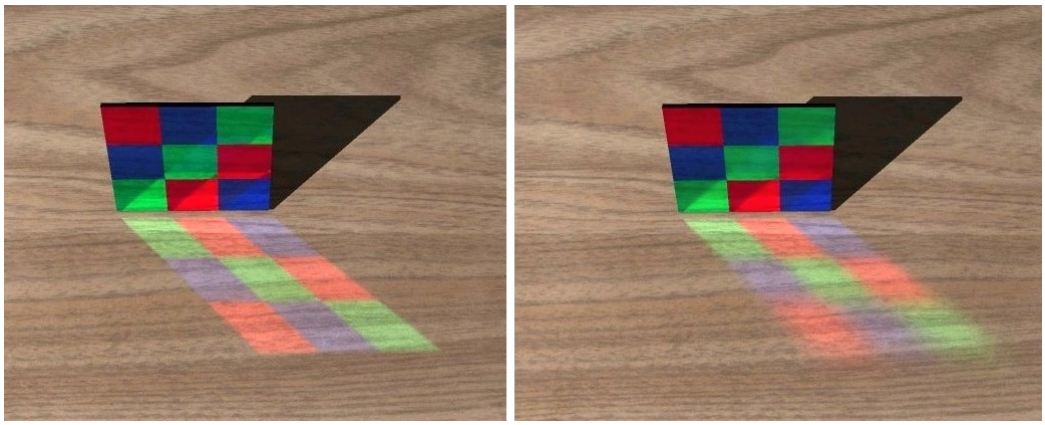


Figure 6.2: Glossy light beam tracing with 0% and 10% scatter distributions on the surface of the tricolour reflector.

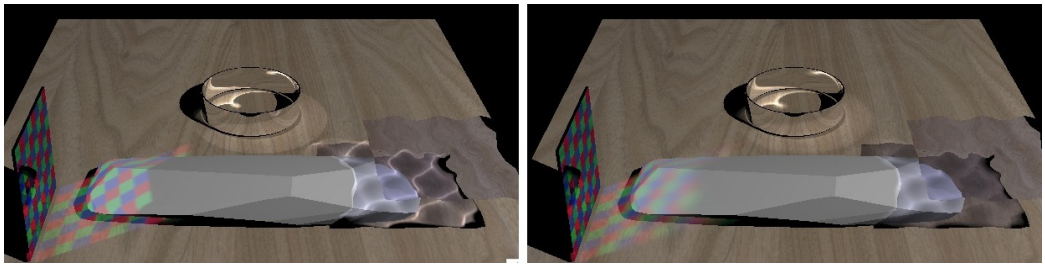


Figure 6.3: Glossy light beam tracing of a more complex scene with 0% and 20% scatter distributions in the left and right images respectively.

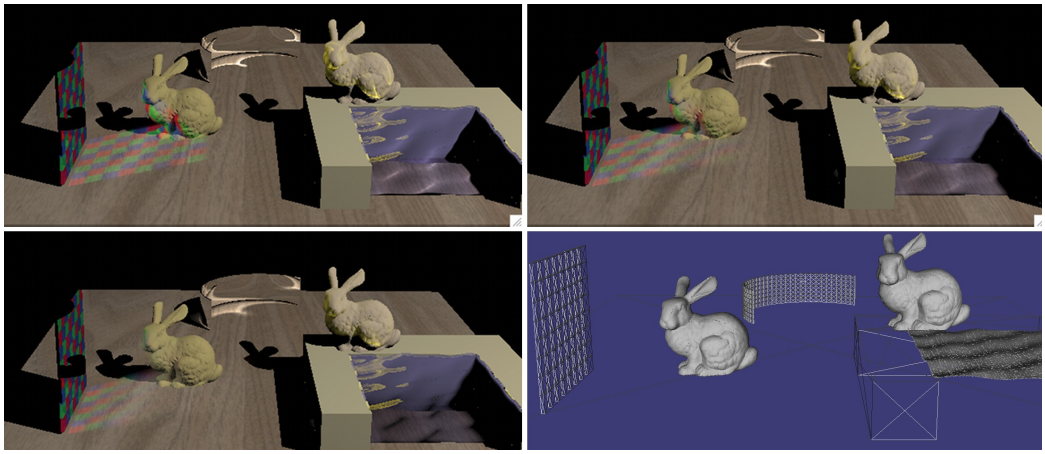


Figure 6.4: Glossy light beam tracing of a more complex scene with a 0%, 10% and 20% scatter distribution in the left, right and bottom right images respectively. The bottom right pane shows a wireframe view of the scene.

rendered in 5.3 seconds, 7.9 seconds and 12 seconds for the 0%, 10% and 20% scatter distributions respectively. Note the change in the execution performance of the beam processing due to the widening of the glossy beams. The bottom right pane in the figure shows a wireframe view of the scene. The Stanford Bunny is the most complex with 65k polygons, the caustic ring has 210 polygons on the inside and the water surface has 4096 polygons. The rendering time for Figure 6.4 is similar to that of Figure 6.3 due to the hierarchical acceleration structures used for beam processing and scene tree-traversal.

In Figure 6.5 LBT (left) is compared to a reference photon map implementation (right) with a scatter lobe standard deviation of 0%. The difference image (bottom, centre) is shown at the same brightness scale. Note that only the $L(S|G)DE$ transport paths are rendered. The ring object (see Figure 6.1) is constructed out of 75×5 scattering surfaces on the inside and also on the outside and renders in 1.5 seconds using LBT and approximately 60 seconds using PM. It takes approximately 20 seconds to radiate the photons and build the kd-tree and 40 seconds to then do the forward render of the image. Rendering the ring object without the caustic $L(S|G)DE$ transport paths takes 0.5 seconds.

The reference images are rendered with 420k photons radiated in the direction of the ring and 100k scattered photons absorbed into the caustic photon map. From the view point shown most of the render time is spent doing the kNN queries. The caustic photon map is rendered directly and a cone filter is applied in the radiance estimate during forward rendering.

Figure 6.6 shows the comparative results for a scatter distribution standard deviation of 10.0% rendered in 2.5 seconds and 60 seconds for LBT and PM respectively. Visually there is again a fairly good match between the glossy LBT and PM results. Figure 6.7 shows the comparative results for a scatter distribution standard deviation of 30.0% rendered in 5.0 seconds and 60 seconds for LBT and PM respectively. Again, also note the change in the execution performance of the beam processing due to the widening of the glossy beams.

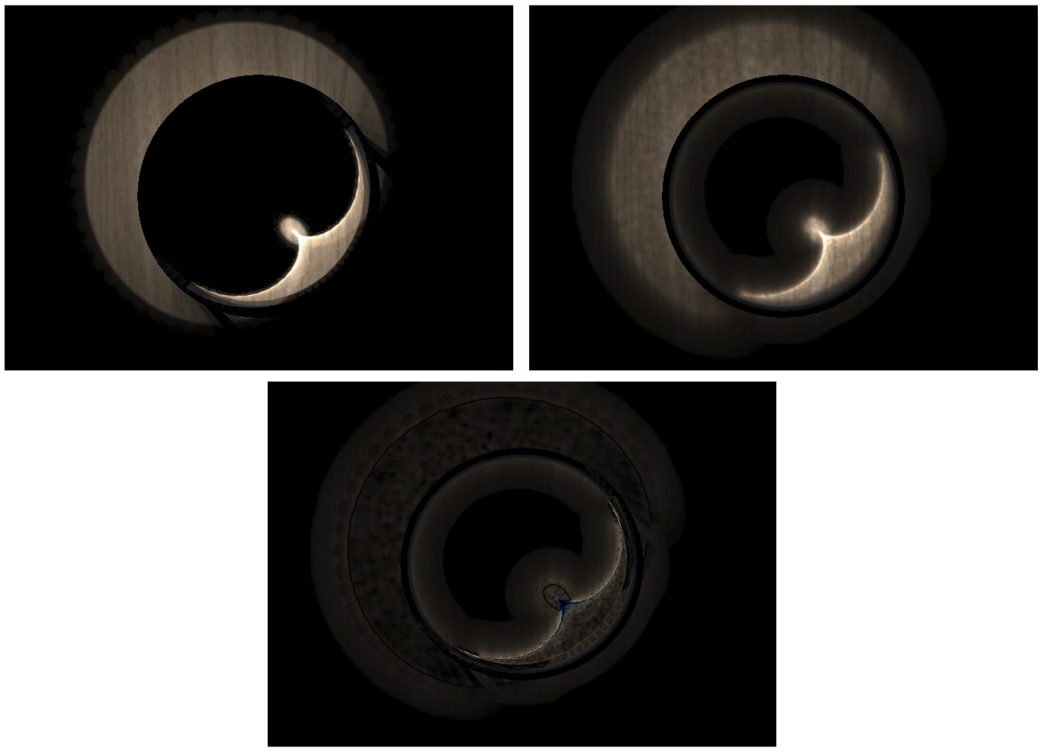


Figure 6.5: Light beam tracing (left) compared to photon mapping (right) of a ring object (Figure 6.1) with a scatter distribution variance of 0; only the $L(S|G)DE$ transport paths are rendered. The bottom image is a difference image.

6.2 Analysis

The appearance of the caustics and the beam shape seem as expected. The difference images also show that there is a fairly good match between LBT and PM. PM is itself a biased method so a small difference between LBT and PM is acceptable. A more detailed comparison of the LBT and light tracing will be done in Part IV of the thesis once further argument in support of the conjecture has been given.

Using PM it takes approximately 60 seconds to render the ring scene whereas the beam tracer renders it in 1.5 to 5 seconds depending on beam width. This is of course because of the forward renderer having to process a hierarchy of only 375 beams to light a fragment instead of doing a radiance estimate (kNN query) from a kd-tree with 100k photons. The proof-of-concept

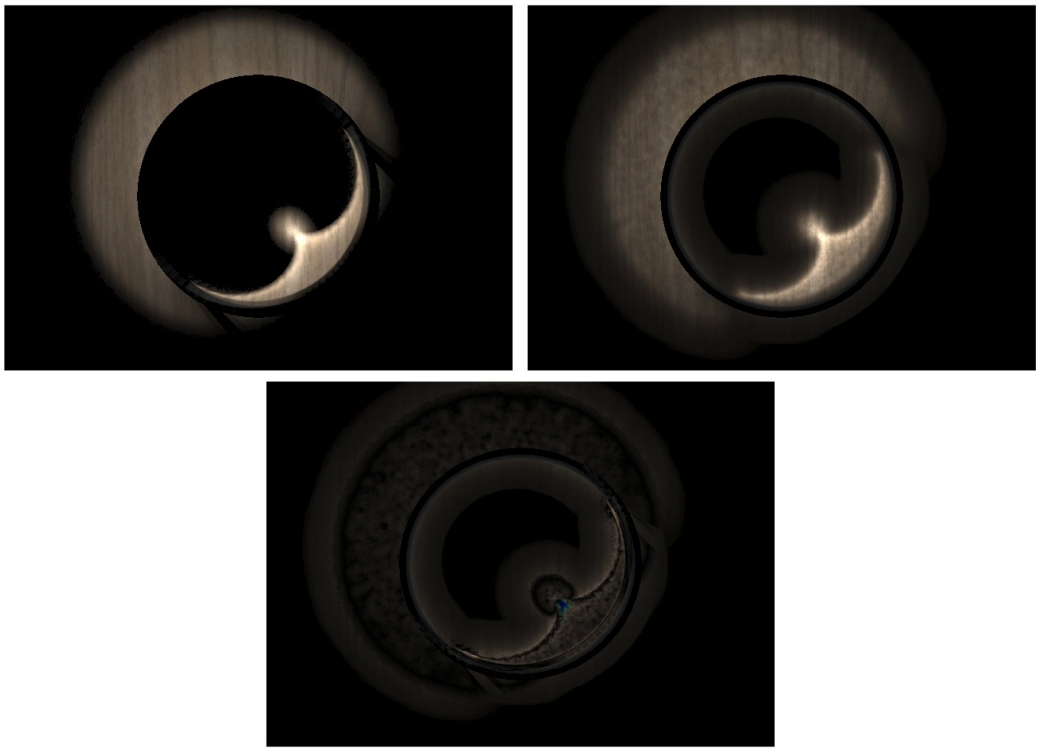


Figure 6.6: Light beam tracing (left) compared to photon mapping (right) of a ring object (Figure 6.1) with a scatter distribution standard deviation of 10%; only the $L(S|G)DE$ transport paths are rendered. The bottom image is a difference image.

PM implementation used for the comparison could only do 8000 kNN queries per second, but lighting a scene using 375 beams instead of 100k photons is certainly a benefit of LBT.

The proposed glossy LBT is able to render single scatter caustics of a quality approaching that of caustic PM. However, three orders of magnitude fewer beams than photons are required causing LBT to be more efficient. It is therefore fair to state that the implementation is a successful proof of concept for glossy LBT.

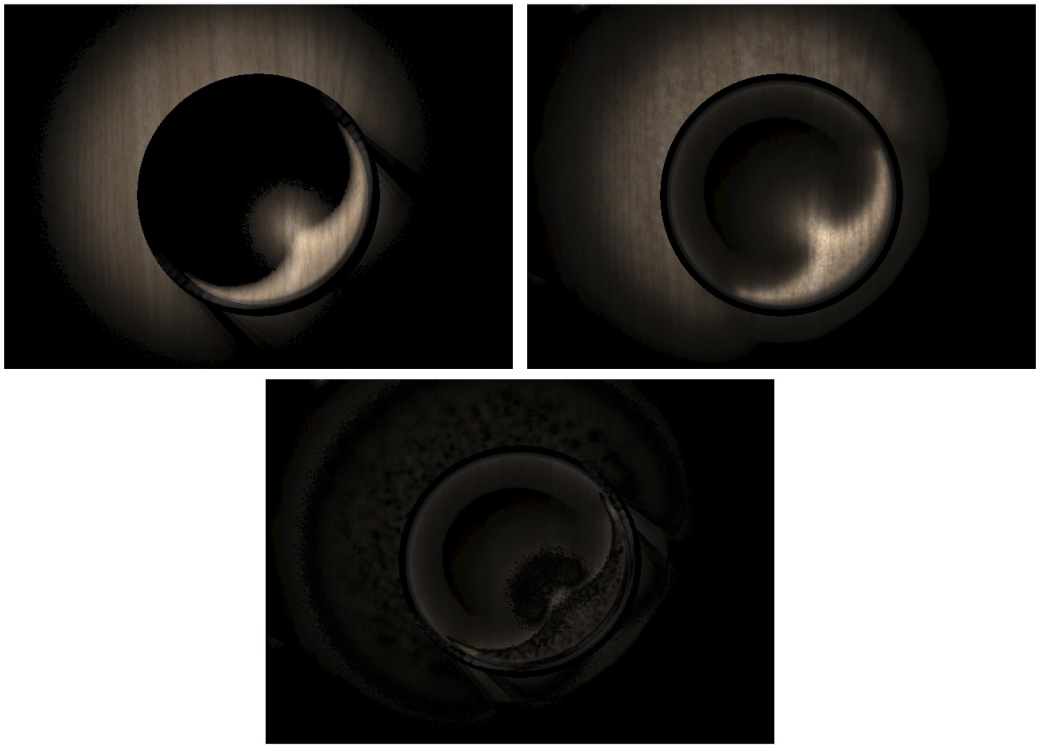


Figure 6.7: Light beam tracing (left) compared to photon mapping (right) of a ring object (Figure 6.1) with a scatter distribution standard deviation of 30%; only the $L(S|G)DE$ transport paths are rendered. The bottom image is a difference image.

Chapter 7

Summary

This chapter summarises the proof of concept of a glossy LBT rendering algorithm. I have extended classical backward (light) beam tracing to include single bounce *glossy* interactions. The algorithm is premised on the conjecture that the lighting integral can be expressed as a function of the volume under a spherical Gaussian probability distribution. The irradiance estimate based on this conjecture is given in Equation 5.1. Three orders of magnitude fewer beams than photons are required, causing LBT to be more efficient for rendering LGDE transport paths.

Given a beam, Equations 5.3 to 5.8 describe how to calculate the $\vec{\phi}_i$ vectors required for the irradiance equation. Then, to calculate the irradiance from a lookup table, one uses:

$$E(x) = \frac{\Phi_s}{A_\perp} \cos\theta \left(\|\vec{\nu}(\vec{\phi}_0, \vec{\phi}_1) + \vec{\nu}(\vec{\phi}_1, \vec{\phi}_2) + \vec{\nu}(\vec{\phi}_2, \vec{\phi}_0)\| \right)$$

and

$$\vec{\nu}(\vec{\phi}_i, \vec{\phi}_j) = \nu'_{Table}(\frac{\phi_i}{\sigma}, \frac{\phi_j}{\sigma}, \theta_{ij}) \frac{\vec{\phi}_i \times \vec{\phi}_j}{\|\vec{\phi}_i \times \vec{\phi}_j\|}$$

ν'_{Table} is the lookup table function for the volume under the spherical Gaussian distribution. The values in the lookup table are derived from a distribution that has a standard deviation of one and mean of zero, requiring

that, before doing the lookups, the lengths ϕ_i and ϕ_j be normalised with σ, the standard deviation of the beam's SG distribution.

The limits and resolution of the lookup table affect the quality of result. I used a table of $512 \times 512 \times 256$ entries representing a domain of 7.5×7.5 standard deviations and $90°$. Inside the code that implements the lookup table a $\theta > 90°$ is split into two table lookups of $\frac{\theta}{2}$ each, as described at the end of Chapter 5.

This generalised LBT rendering algorithm now provides a lumped[1] model of L(S|G)DE transport paths. It allows simulation of glossy caustic transport paths at an order of a magnitude faster than rendering algorithms such as PM and light tracing. Note that the irradiance calculation at x assumes that there are no occlusions between the scattering surface and x. Section 12.4 discusses shadowing of the beam flux.

Based on the conjecture and the successful proof of concept, the rest of the thesis attempts to formally derive the glossy beam irradiance equation and further develop the glossy LBT rendering algorithm. The ν'_{Table} lookup table is also replaced with a more accurate solution discussed in Section 12.3. Multi-bounce glossy interactions are implemented in Chapter 13 once the glossy beam irradiance equation (currently based on conjecture) has been derived from the rendering equation.

[1] A lumped element model simplifies the behaviour of a physical system under certain assumptions to a single component or equation.

Part III

The Glossy Scatter Lobe BRDF

This part of the thesis shows the derivation and verification for physical plausibility of the glossy BRDF that is required to support the conjecture given in the previous part, Part II. Most of the information regarding the numerical verification was originally published in a research paper [56].

Chapter 8

Derivation of the Glossy BRDF

This chapter shows the derivation of the glossy BRDF that is required to support the conjecture presented in the previous part, Part II. The next two chapters then numerically verify that the BRDF is physically plausible.

8.1 Overview

The proposed glossy beam irradiance equation, Equation 5.1, is repeated below for easy reference:

$$E(x) = \frac{\Phi_s}{A_\perp} |\cos(N_x, \overline{\Psi})| \iint_\Omega \rho(\vec{\phi}) d\omega_\phi. \qquad (5.1)$$

Remember that Equation 5.1 is a conjecture presented in the previous part, Part II, of the thesis. The plausibility of the conjecture rests in part on the existence of a glossy BRDF that results in a spherical Gaussian scatter distribution. The suppositions investigated here are that such a BRDF can be formulated and that it is physically plausible.

Usually the BRDF at y would be defined as:

$$\begin{aligned} f_r(y, \Psi' \rightarrow -\Psi) &= \frac{dL(y \rightarrow -\Psi)}{dE(y \leftarrow \Psi')} \\ &= \frac{dL(y \rightarrow -\Psi)}{L(y \leftarrow \Psi')|\cos(N_y, \Psi')| d\omega_{\Psi'}} \end{aligned}$$

Recall that $dE(y \leftarrow \Psi')$ is the differential irradiance at y due to a differential solid angle $d\omega_{\Psi'}$ around the incoming direction Ψ' and $dL(y \rightarrow -\Psi)$ is the differential radiance leaving in direction $-\Psi$.

To start off the derivation of the glossy BRDF, note that the photon probability distribution function is:

$$P_r(y, \Psi' \rightarrow -\Psi) = f_r(y, \Psi' \rightarrow -\Psi)|\cos(N_y, -\Psi)|$$

The probability volume above the surface and within this probability distribution function is in general *smaller* than one. Some light is not scattered to the hemisphere above the surface, but rather absorbed into the surface.

Further, for a specular or glossy BRDF the incident and exitant angles are approximately equal, allowing one to write:

$$\begin{aligned} P_r(y, \Psi' \rightarrow -\Psi) &\approx f_r(y, \Psi' \rightarrow -\Psi)|\cos(N_y, \Psi')| \\ &= \left(\frac{dL(y \rightarrow -\Psi)}{L(y \leftarrow \Psi')|\cos(N_y, \Psi')|d\omega_{\Psi'}}\right)|\cos(N_y, \Psi')| \\ &= \frac{dL(y \rightarrow -\Psi)}{L(y \leftarrow \Psi')d\omega_{\Psi'}} \end{aligned} \tag{8.1}$$

One of the conditions for physical plausibility of a BRDF is symmetricity. However, this approximation will result in a slight BRDF asymmetry. The asymmetry, which is discussed further in the next two chapters, is negligible for specular and glossy BRDFs. Also, Equation 8.1 leads to:

$$f_r(y, \Psi' \rightarrow -\Psi) \approx \frac{P_r(y, \Psi' \rightarrow -\Psi)}{|\cos(N_y, \Psi')|} \approx \frac{dL(y \rightarrow -\Psi)}{L(y \leftarrow \Psi')d\omega_{\Psi'}|\cos(N_y, \Psi')|} \tag{8.2}$$

8.2 Derivation

This section gives the argument for the conjecture that a glossy BRDF with a spherical Gaussian scatter lobe is physically plausible. I propose that a glossy BRDF with a negligible diffuse component be recast using the scatter

direction offset $\vec{\phi}$ described in Equation 5.2. The BRDF then becomes:

$$f_r(y, \Psi' \rightarrow -\Psi) \quad = \quad f_r(y, \Psi', \vec{\phi})$$

Recall that $\vec{\phi}$ is the 2D (e.g. azimuth and zenith angles) offset of the outgoing direction $-\Psi$ relative to the specular scatter direction. This is similar to the formulation of the original Phong [11] reflection model and the modified Phong reflection model used by Lafortune and Williams [57] for physically based rendering.

Using Equation 5.2 and Equation 8.1, the glossy BRDF becomes:

$$\begin{aligned}
f_r(y, \Psi', \vec{\phi}) &\approx \frac{dL(y \rightarrow R + \vec{\phi})}{L(y \leftarrow \Psi')|\cos(N_y, \Psi')|d\omega_{\Psi'}} \\
&= \left[\frac{dL(y \rightarrow R + \vec{\phi})}{L(y \leftarrow \Psi')}\frac{1}{d\omega_{\Psi'}}\right]\frac{1}{|\cos(N_y, \Psi')|} \\
&= \left[P(\vec{\phi})\right]\frac{1}{|\cos(N_y, \Psi')|} \\
&= \left[\rho_s\rho(\vec{\phi})\right]\frac{1}{|\cos(N_y, \Psi')|} \qquad (8.3) \\
&= \left[\rho_s\frac{1}{2\pi\sigma^2}e^{-\frac{\phi^2}{2\sigma^2}}\right]\frac{1}{|\cos(N_y, \Psi')|} \qquad (8.4)
\end{aligned}$$

The definition of ρ is therefore:

$$\rho_s\rho(\vec{\phi}) \quad = \quad P(\vec{\phi}) = \left(\frac{dL(y \rightarrow R + \vec{\phi})}{L(y \leftarrow \Psi')}\frac{1}{d\omega_{\Psi'}}\right) \qquad (8.5)$$

As is the case for the Phong BRDF, although R is dependent on the incoming direction Ψ', the normalised *shape* of the scatter lobe $P(\vec{\phi})$ is by definition independent of the incident direction Ψ'. By defining the BRDF in this way, $\rho(\vec{\phi})$ is the PDF of the *scattered* light rays around R. Specular reflectivity due to material properties and Fresnel effects are embedded in the specular coefficient ρ_s.

As the standard deviation of the spherical Gaussian approaches zero $\rho(\vec{\phi})$ becomes increasingly similar to the Dirac delta function of the specular

BRDF (defined in Equation 2.5). In the limit the Glossy BRDF becomes equal to the specular BRDF:

$$\lim_{\sigma \to 0} \frac{\rho_s \rho(\vec{\phi})}{|\cos(N_y, \Psi')|} = \frac{\rho_s \delta(\vec{\phi})}{|\cos(N_y, \Psi')|}$$

Practically the BRDF also works well for Monte Carlo transport simulations. If ρ is a Gaussian PDF, then a scatter direction can be chosen from a Gaussian random vector distribution for importance sampling and the BRDF may be sampled directly using rejection sampling for a given $(y, \Psi_i \to \Psi_o)$. The software code for generating random vectors with various PDFs is discussed in more detail later in Part V of the thesis.

8.3 Physical Plausibility

A BRDF may be considered *physically plausible* if three conditions are met. These conditions are:

- Positivity.
- Symmetricity.
- Conservation of energy.

In the following subsections these conditions are explained and tested analytically against the glossy BRDF.

8.3.1 Positivity

Positivity states that $f_r(\Psi_i \to \Psi_o) \geq 0$ for all Ψ_i and Ψ_o. In other words, an object can never have a negative reflectance.

The glossy BRDF defined in Equation 8.3 is positive if the specular reflectivity ρ_s is positive which is always the case.

8.3.2 Symmetricity

Symmetricity implies that the surface reflectance stays unmodified should the direction of the light be reversed viz. $f_r(\Psi_i \rightarrow \Psi_o) = f_r(\Psi_o \rightarrow \Psi_i)$. In other words, f_r is unmodified should the source and receiver positions be swapped around. This is sometimes referred to as BRDF reciprocity, as it is related to the Helmholtz reciprocity principle as explained by Veach [9].

Manipulating the left and right sides of the symmetricity condition using the definition of the glossy BRDF results in:

$$\begin{aligned} f_r(y, \Psi_i \rightarrow \Psi_o) &= f_r(y, \Psi_i \leftarrow \Psi_o) \\ f_r(y, \Psi_i \rightarrow (R_{\Psi_i} + (\Psi_o - R_{\Psi_i}))) &= f_r(y, (R_{\Psi_o} + (\Psi_i - R_{\Psi_o})) \leftarrow \Psi_o) \\ f_r(y, \Psi_i, (\Psi_o - R_{\Psi_i})) &= f_r(y, \Psi_o, (\Psi_i - R_{\Psi_o})) \\ \rho_s \rho(\Psi_o - R_{\Psi_i}) \frac{1}{|\cos(N_y, \Psi_i)|} &= \rho_s \rho(\Psi_i - R_{\Psi_o}) \frac{1}{|\cos(N_y, \Psi_o)|} \end{aligned} \tag{8.6}$$

It can easily be shown from the mirror-like specular geometry that the length of $\Psi_o - R_{\Psi_i}$ is the same as the length of $\Psi_i - R_{\Psi_o}$. If $\rho(\vec{\phi})$ is symmetric around R at $\phi = 0$ then the BRDF obeys symmetricity when $\frac{1}{|\cos(N_y, \Psi_i)|} \approx \frac{1}{|\cos(N_y, \Psi_o)|}$. In other words the defined BRDF is symmetric close to the specular scatter geometry where the glossy BRDF is significant.

8.3.3 Conservation of Energy

The principle of conservation of energy states that light energy can only be absorbed or scattered. In other words, the exitant radiant power per unit area (i.e. radiosity) leaving a surface cannot be more than the incident radiant power (i.e. irradiance) on the surface. The differential radiosity, $dB(x)$ due to the irradiance $dE(x \leftarrow \Psi_i)$ that is due to the incoming radiance $L(x \leftarrow \Psi_i)$ from a differential solid angle $d\omega_{\Psi_i}$ around the incoming direction Ψ_i may be

calculated as:

$$\begin{aligned} dB(x) &= \int_{\Omega_o} dB(x \rightarrow \Psi_o) \\ &= \int_{\Omega_o} [dL(x \rightarrow \Psi_o)] \,|\cos(N_x, \Psi_o)| d\omega_{\Psi_o} \\ &\leq dE(x \leftarrow \Psi_i) \end{aligned} \tag{8.7}$$

Equation 8.7 must hold for all Ψ_i. Simplifying using the definition of the BRDF, one arrives at:

$$\begin{aligned} \int_{\Omega_o} [f_r(x, \Psi_i \rightarrow \Psi_o) dE(x \leftarrow \Psi_i)] \,|\cos(N_x, \Psi_o)| d\omega_{\Psi_o} \\ \leq dE(x \leftarrow \Psi_i) \quad \forall\ \Psi_i \\ \therefore \int_{\Omega_o} [f_r(x, \Psi_i \rightarrow \Psi_o)] \,|\cos(N_x, \Psi_o)| d\omega_{\Psi_o} \leq 1 \quad \forall\ \Psi_i \end{aligned} \tag{8.8}$$

Manipulating the energy conservation condition using the definition of the glossy BRDF results in:

$$\int_{\Omega_o} f_r\left(y, \Psi_i \rightarrow \Psi_o\right) |\cos\left(N_y, \Psi_o\right)| d\omega_o \tag{8.9}$$

$$\begin{aligned} &= \int_{\Omega_{\Psi_o}} \rho_s \rho(\vec{\phi}) \frac{1}{|\cos(N_y, \Psi_i)|} |\cos\left(N_y, \Psi_o\right)| d\omega_{\Psi_o} \\ &= \frac{\rho_s}{|\cos(N_y, \Psi_i)|} \int_{\Omega_{\Psi_o}} \rho(\vec{\phi}) |\cos\left(N_y, \Psi_o\right)| d\omega_{\Psi_o} \\ &= \frac{\rho_s}{|\cos(N_y, \Psi_i)|} \int_{\Omega_{\Psi_o}} \rho(\vec{\phi}) |\cos\left(N_y, R_{\Psi_i} + \vec{\phi}\right)| d\omega_{\Psi_o} \\ &\approx \frac{\rho_s}{|\cos(N_y, \Psi_i)|} |\cos\left(N_y, R_{\Psi_i}\right)| \text{ for sharp } \rho(\vec{\phi}) \end{aligned} \tag{8.10}$$

$$\begin{aligned} &= \frac{\rho_s}{|\cos(N_y, \Psi_i)|} |\cos\left(N_y, \Psi_i\right)| \\ &= \rho_s \end{aligned} \tag{8.11}$$

The total reflectivity is approximately the specular reflectance ρ_s which is always smaller than one.

Chapter 9

Numerical Verification of the Glossy BRDF for Physical Plausibility

This chapter presents a suite of numerical tests to verify that a BRDF implementation is physically plausible. The glossy BRDF derived in the previous chapter as well as two other well known BRDFs are verified for physical plausibility in the next chapter. Verifying physical plausibility of material BRDF implementations is a critical requirement for comparing different rendering algorithms.

The role and value of verification (of implementations) and qualification or validation (of the BRDF expressions) in the modelling and simulation process is discussed in detail by authors such as Sargent [58][59] and Zeigler et al. [60]. Figure 9.1 shows the modelling process. The verification step is the bottom arc in the process which is part of the *computer programming and implementation* step.

This chapter focusses on the verification (of the implementation) of material BRDFs against the conceptual model of physical plausibility as opposed to algebraically proving that the BRDFs are physically plausible.

Aspects such as symmetricity and conservation of energy of each BRDF are verified by appropriately evaluating the implementation on a large set of

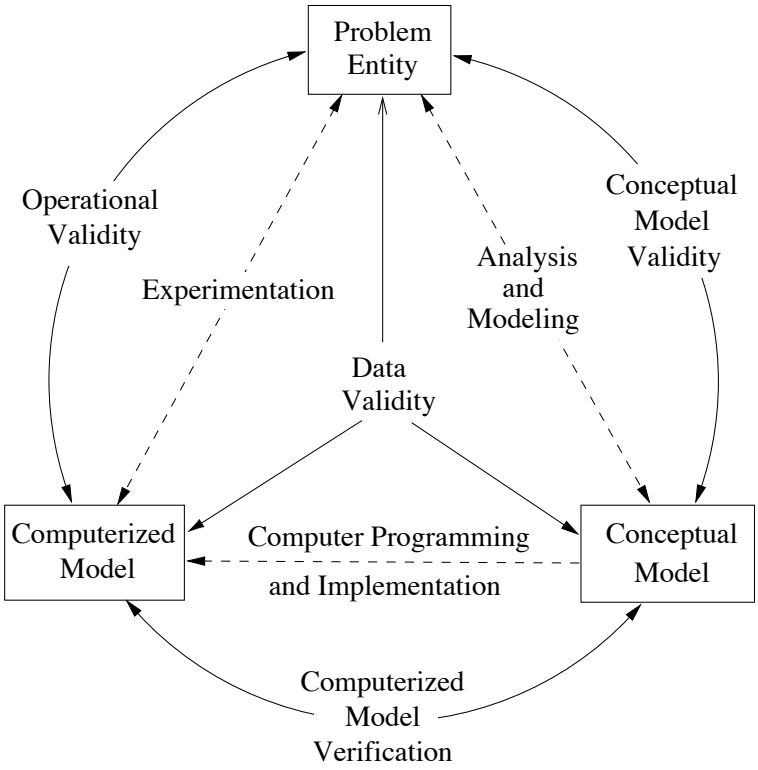


Figure 9.1: The modelling process, adapted from Sargent [59].

equidistant input vectors. Numerical methods such as discussed in this chapter are useful for verifying the correctness of the implementation of derived as well as measured BRDFs. As far as I am aware, numerical verification of the physical plausibility of BRDF implementations has not previously been shown in any published works.

A total of five tests are to be outlined in this chapter, and the results of applying these tests to each of 3 BDRFs are presented in the next chapter.

Firstly, the quality of the set of equidistant vectors is analysed. A good set of vector bins is crucial for an accurate BRDF analysis. This matters is discussed in Section 9.1.

Secondly, because the BRDF implementations shown here make use of random vector PDFs as the basis for generating random scatter directions, the PDFs are tested for proper normalisation in isolation from the BRDFs.

The test for proper normalisation is discussed in Section 9.2.

Thirdly, each BRDF implementation is compared to its own scatter implementation. The value of f_r is compared to:

- A BRDF lookup table generated by binning random vectors that are the result of rejection sampling against f_r.
- A BRDF lookup table generated by binning random scatter directions generated by the BRDF implementation.

This test is discussed in more detail in Section 9.3.

Finally, each BRDF implementation is analysed for symmetricity and energy conservation. These tests are discussed in Section 9.4 and Section 9.5 respectively.

9.1 Generating Equidistant Input Vectors

Good quality equidistant direction vectors[1] are often required. One use of such vectors that are important here are for bin directions when accumulating PDFs and doing numerical integrals on the sphere or hemisphere. Marques et al. [61] describe the importance of good quality equidistant vectors in rendering in general. They demonstrate a marked improvement in their simulation results when using good vector distributions.

The method chosen to generate equidistant vectors starts with an icosahedron (polyhedron with 20 triangular faces) and over a number of iterations subdivides every triangle into four similar triangles until at least the required number of vertices have been generated.

After each subdivision all vertices are normalised i.e. pushed out onto the unit sphere. Figure 9.2 shows such a subdivision of an icosahedron. The resulting polyhedron is also referred to as a geodesic dome. Due to the processing of duplicate vertices the generation of equidistant vectors has a *measured* average computational complexity of $\mathcal{O}(\log n)$ for n the number of required equidistant vectors.

[1] Vectors to points that are equidistant on the unit sphere.

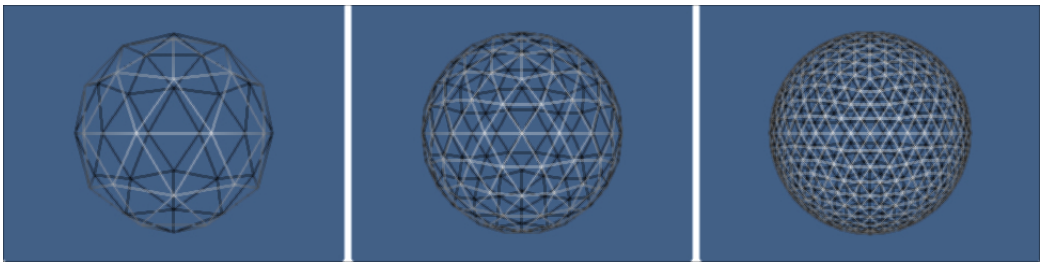


Figure 9.2: Icosahedron-based subdivision with 42, 162 and 642 vectors.

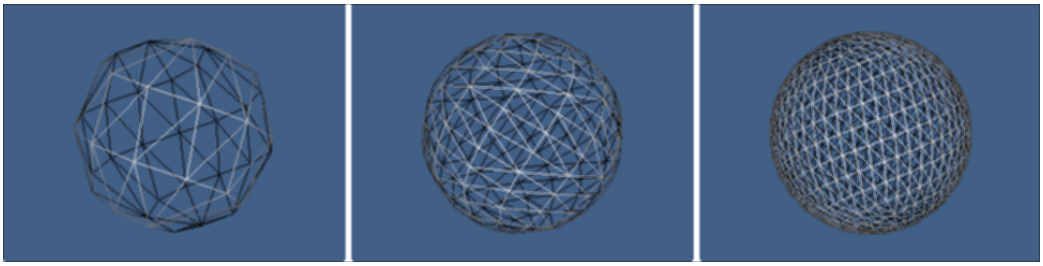


Figure 9.3: Fibonacci-spiral-sphere-based subdivision with 42, 162 and 642 vectors.

Other methods to generate equidistant vectors may also be used. Figure 9.3, for example, shows the result of the Fibonacci spiral sphere (also known as the golden section spiral) method used by Marques et al. [61]. The polyhedron subdivision method is preferred, however, because the vertices are explicitly connected into faces during their creation. Calculating the mesh connectivity on a set of n already existing vertices is an expensive $\mathcal{O}(n \log n)$ operation and the resulting mesh quality may not be assured.

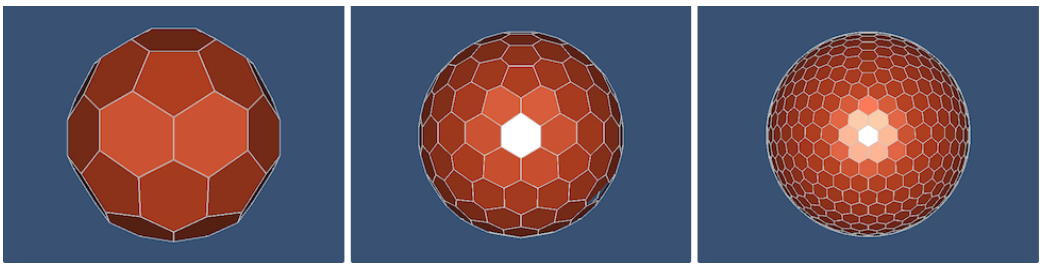


Figure 9.4: Geometric dual to show the bin shape of Icosahedron-based subdivision with 42, 162 and 642 vectors.

As mentioned the equidistant vectors will be used as bins to accumulate

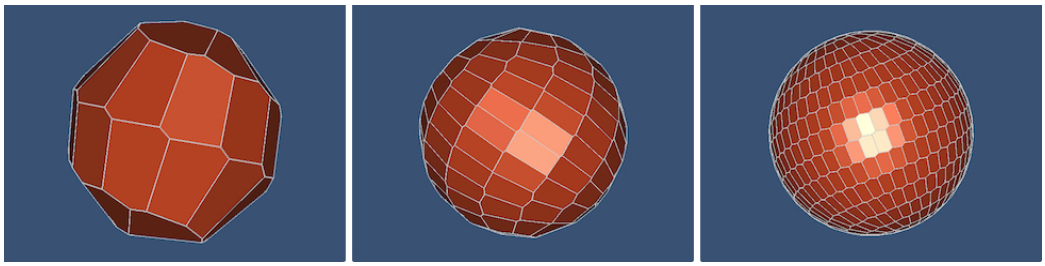


Figure 9.5: Geometric dual to show the bin shape of Fibonacci-spiral-sphere-based subdivision with 42, 162 and 642 vectors.

random vectors. Figure 9.4 and Figure 9.5 show what the Voronoi bins of the geodesic dome and Fibonacci Spiral Sphere look like.

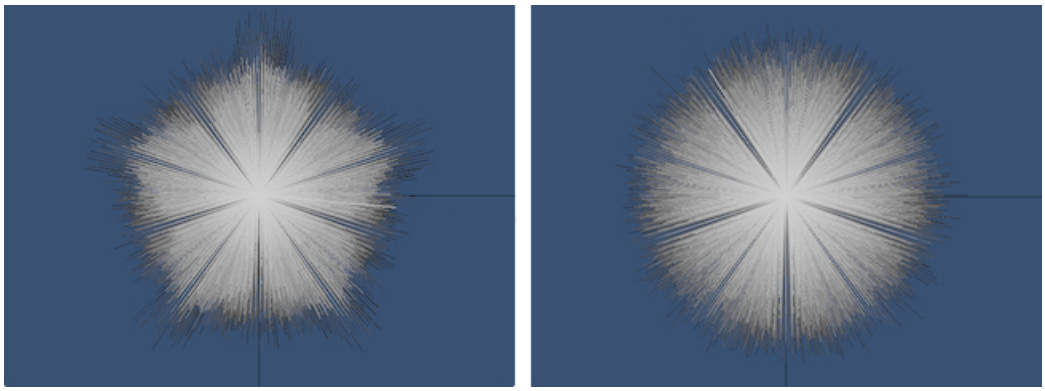


Figure 9.6: Top View of the rejection sampled diffuse BRDF for unrelaxed bins on the left and relaxed bins on the right. Note the improvement in the quality of the binned dome like BRDF.

Figure 9.5 shows that the initial bins are only quasi-regular. The subdivided mesh should be processed to relax the *tension* created by the variance in the local vertex density. Figure 9.6 shows the result of binning random unit vectors using an unrelaxed bin vector set on the left vs. a relaxed bin vector set on the right. The bin values are generated by binning the random unit vectors into the closest (i.e. Voronoi) bin. One should be seeing a dome-like distribution. The five wide and five narrow gaps visible in the figures are the gaps between the geodesics on the dome.

One iteration of relaxation proceeds as shown in Figure 9.7. Making k equal to 1% of the total number of vectors seems to work well. s is related

Figure 9.7 Pseudocode for one iteration of bin relaxation.

```
/*! Relax bins using k-nearest neighbours and
    relaxation speed s.*/
void relaxBins(set of bins, k, s)
{
  for each vector V in the set of bins
  {
    neighbours = k-nearest neighbours to V;
    maxRadiusSq = square of radius of neighbours;
    minRadius = radius of nearest neighbour;

    for each vector U in neighbours
    {
      weight = 1.0 - (V - U).lengthSq()/maxRadiusSq;
      direction = (V - U).normalised();
      relaxVector += direction * weight;
    }

    V+=relaxVector * (minRadius * s);
  }
}
```

to the relaxation speed and a value of 0.1 or 10% was used. Seven to ten relaxation iterations seems to work well. Ten relaxation iterations were used for the results shown here for 10242 vectors.

It is possible to over-relax the bins such that the original vertex connectivity becomes stale. Future work could focus on analysing the bin shape and size during relaxation to detect when to automatically stop the relaxation.

The variance for the icosahedron based method is naturally higher than the variance for the Fibonacci spiral sphere method. However, after relaxation the variances become similar. Relaxing 10242 vectors currently takes about 8 seconds on a Core 2 Duo 2.0GHz CPU core. This is still significantly faster than using the Fibonacci spiral sphere method combined with a post-process to calculate the mesh connectivity.

9.2 Verifying the Random Vector PDFs

The BRDF scatter functions are based on random vector PDFs. Like all PDFs an integration of the probability over the function domain must integrate to one. To test this property the integration is executed numerically over the hemispherical vector domain.

Firstly the set of n random vectors are generated by calling the appropriate PDF method n times. Calling `Vec3::randGaussianLobe(const float sd)`, for example returns a random vector of a spherical Gaussian distribution with mean the vector $(0, 0, 1)$ and a standard deviation of sd. The generated vectors are accumulated in the set of equidistant bins already generated. As a rule of thumb the number of random vectors that are generated is a hundred times greater than the number of bins.

The numerical integration is then executed over the accumulated probabilities in the bins. A second numerical integration is executed over the probabilities returned from the PDF implementation. Both integrals should evaluate to one and the per bin difference between the accumulated and directly calculated PDFs should be zero. The pseudocode for verifying a random vector PDF is shown in Figure 9.8. `binSR` is the solid angle in steradian subtended by a bin.

Figure 9.8 Pseudocode for verifying a random vector PDF.

```
/*! Verify PDF using equidistant set of bins */
void VerifyPDF(set of bins, PDF)
{
  numPDFSamples = 100 x number of bins
  binSR=(4 * PI) / (number of bins)

  for numPDFSamples
  {
    Query the PDF implementation for a random vector;
    Accumulate vector in appropriate Voronoi bin;
  }

  binSum = 0;
  directSum = 0;
  error = 0;

  for each vector V in the set of bins
  {
    binProb = binValue for V / numPDFSamples;
    directProb = PDF(V) * binSR;

    binSum += binProb;
    directSum += directProb;
    error += fabs(binProb - directProb);
  }

  Report binSum, directSum and error;
  //Sums should be one.
  //Error should be zero.
}
```

9.3 Verifying the BRDF Against Its Photon Scatter Operators

As mentioned in the introduction to this section, the value of f_r is compared to:

- A BRDF lookup table generated by binning random vectors that are the result of rejection sampling against f_r.
- A BRDF lookup table generated by binning random scatter directions generated by the BRDF implementation.

The test proceeds as shown in Figure 9.9. The results are shown in Chapter 10 under the table entry *Error between f_r and the direct scatter BRDF* and *Error between f_r and the rejection sampling BRDF*. The direct scatter BRDF and the rejection sampled BRDF are calculated from the binned samples while f_r is an evaluation of the BRDF expression. The error results are the absolute difference in the energy conservation performance of the BRDF expression and the BRDF calculated from the binned samples (a value between zero and one) expressed as a percentage.

Figure 9.9 Pseudocode for verifying the BRDF against its photon scatter operators.

```
void VerifyBRDFImpl(set of bins, BRDF)
{
  binSR=(4 * PI) / (number of bins)

  for each vector A in the set of bins
  {
    numPDFSamples = 500 x number of bins
    setup temporary PDF_reject_samp bin set
    setup temporary PDF_direct bin set

    error_reject_samp = 0;
    error_direct = 0;

    for numPDFSamples
    {
      scatDir_reject_samp = BRDF.scatter_reject_samp(A);
      scatDir_direct = BRDF.scatter_direct(A);

      Accumulate scatDir_reject_samp in PDF_reject_samp;
      Accumulate scatDir_direct in PDF_direct;
    }

    for each vector B in the set of bins
    {
      PDF_reject_samp = ((PDF_reject_samp[B].value() /
                numPDFSamples);
      PDF_direct = ((PDF_direct[B].value() /
                numPDFSamples);

      BRDF = BRDF(A, B);
      cosTheta=B*normal;

      error_reject_samp +=
        fabs(BRDF*binSR*cosTheta-PDF_reject_samp);
      error_direct +=
        fabs(BRDF*binSR*cosTheta-PDF_direct);
    }
    Keep track of min, max and average error values;
  }
  Report min, max and average error statistics;
}
```

9.4 Verifying BRDF Symmetry

Figure 9.10 Pseudocode for verifying BRDF symmetry.

```
void VerifyBRDFSymmetry(set of bins, BRDF)
{
  binSR=(4 * PI) / (number of bins)

  for each vector A in the set of bins
  {
    error=0;

    for each vector B in the set of bins
    {
      BRDF_forward=BRDF(A, B);
      BRDF_backward=BRDF(B, A);

      cosTheta=B*normal;

      error +=
        fabs((BRDF_forward-BRDF_backward)*
        binSR*cosTheta);
    }
    Keep track of min, max and average error values.
  }
  Report min, max and average error statistics;
}
```

The symmetry condition

$$f_r(\Psi_i \rightarrow \Psi_o) = f_r(\Psi_o \rightarrow \Psi_i)$$

is numerically evaluated using the set of input vectors. The pseudocode is shown in Figure 9.10. The result of this test is shown in Chapter 10 under the table entry *BRDF symmetry*. The values indicate the fraction of energy lost due to symmetry error.

9.5 Verifying BRDF Energy Conservation

Figure 9.11 Pseudocode for verifying the BRDF energy conservation.

```
/*! Verify BRDF energy conservation using equidistant set of bins */
void VerifyBRDFEnergyCons(set of bins, BRDF)
{
  binSR=(4 * PI) / (number of bins)

  for each vector A in the set of bins
  {
    result=0;

    for each vector B in the set of bins
    {
      cosTheta=B*normal;

      result+=BRDF(A, B) * (cosTheta * binSR);
    }

    Keep track of min, max and average of conservation result values.
  }

  Report min, max and average of conservation result statistics.
  //The conservation result should be <= 1.0.
}
```

The energy conservation condition

$$\int_{\Omega_o} f_r\left(x, \Psi_i \rightarrow \Psi_o\right) \left|\cos\left(N_x, \Psi_o\right)\right| \, d\omega_o \leq 1 \quad \forall \; \Psi_i$$

is numerically evaluated using the set of input vectors. The pseudocode is shown in Figure 9.11. The result of this test is shown in Chapter 10 under the table entry *Energy conservation*.

Chapter 10

Results and Analysis

This chapter gives the results of the numerical verification of the glossy BRDF derived in Chapter 8. The verified expressions for the diffuse BRDF, the original Phong BRDF as well as the micro-facet based Blinn-Phong BRDF (as derived by Pharr and Humphreys [10]) are also presented.

10.1 Results

The verification results of the two vector distributions are given in Table 10.1. The integration results for a sample set from each of the two random vector distributions are shown. Notice that `binSum` and `directSum` are equal and one indicating that the PDFs are implemented correctly.

Table 10.1: Numerically calculated volumes of random vector PDFs for a cosine lobe exponent s of 10.0 and a Gaussian lobe variance of 0.1.

Distribution	`binSum`	`directSum`	Used By
Cosine ($\cos^s$) lobe	1.00	1.00	Diffuse, Phong, Blinn-Phong
Gaussian lobe	1.00	1.00	Glossy

Although numerical verification of physical plausibility could be executed

on any BRDF, implementations of the following BRDFs were verified for physical plausibility:

- Lambertian diffuse BRDF
- Blinn-Phong BRDF
- Phong BRDF
- Glossy BRDF

The BRDF equations (taken from literature) that were found not to be physically plausible were modified as required. The modifications involved visualisation and metering of the binned test results (which was quite valuable) combined with some experimentation.

These adapted BRDF formulations are presented in the following sections. The Lambertian diffuse BRDF is straightforward to implement and known to be positive, symmetric and energy conserving. It may be used as a simple control for the verification process.

The software code for the BRDFs and numerical verifications may be found at `https://code.google.com/p/stitch-engine/`. The results shown here were generated using tag *sprinkles* of the TestBRDFs software. The software uses the cmake build system. Look for the `TestBRDFs` target in the `CMakeLists.txt` file or the `main_TestBRDFs cpp` source file.

10.2 The Lambertian Diffuse BRDFs

The Lambertian diffuse and specular BRDF expressions as given in the background chapter are quite common and known to be physically plausible. Since the Lambertian diffuse BRDF is known to be energy conserving, symmetric and positive, it may be used as an additional sanity check on the code which checks for physical plausibility. Figure 10.1 shows the resulting diffuse BRDF. Table 10.2 shows the results of the BRDF analysis. The fact that the BRDF analysis very nearly matches the verification targets gives some confidence in the verification process. The remaining 2% of error is assumed

to be due to numerical rounding error and used as a reference for the other BRDFs.

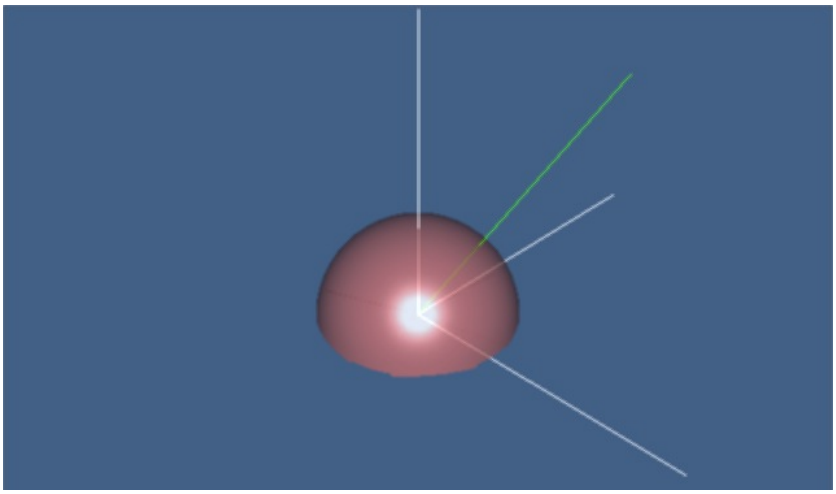


Figure 10.1: The diffuse BRDF.

Table 10.2: Lambertian diffuse BRDF.

Test	Target	Unrelaxed	Relaxed
Error between f_r and the rejection sampling BRDF	0.00%	min=5.99% avrg=6.08% max=6.15%	min=1.95% avrg=2.01% max=2.08%
Error between f_r and the direct scatter BRDF	0.00%	min=6.00% avrg=6.08% max=6.15%	min=1.96% avrg=2.01% max=2.05%
BRDF symmetricity error / fraction of energy lost	0.00	min=0.00 avrg=0.00 max=0.00	min=0.00 avrg=0.00 max=0.00
Energy conservation	≤ 1.00	min=1.00 avrg=1.00 max=1.00	min=1.00 avrg=1.00 max=1.00

10.3 The Blinn-Phong BRDF

The Blinn-Phong [12] BRDF to be verified for physical plausibility is defined as:

$$\begin{aligned} f_r(x, \Psi_i \rightarrow \Psi_o) &= \frac{\rho_d}{\pi} + \frac{\rho_s D(\Psi_H)}{4|\cos(N_x, \Psi_i)||\cos(N_x, \Psi_o)|} \\ D(\Psi_H) &= \left(\frac{s+2}{2\pi}|\cos^s(N_x, \Psi_H)|\right) \end{aligned}$$

Ψ_H is the halfway vector between Ψ_i and Ψ_o. s is the specular exponent of the specular cosine scatter lobe. The Blinn-Phong BRDF has both a Lambertian diffuse reflection component and a specular reflection component. Note that the effect of shadowing and masking of the microfacets is not included in the above definition. Including shadowing and masking in the numerical verification is left for future work.

The formulation found to be physical plausible and given above is the same as the formulation by Pharr and Humphreys [10], but different from the formulation given by Dutré et al. [3]. Figure 10.2 shows the resulting Blinn-Phong BRDF. Table 10.3 shows the results of the BRDF analysis.

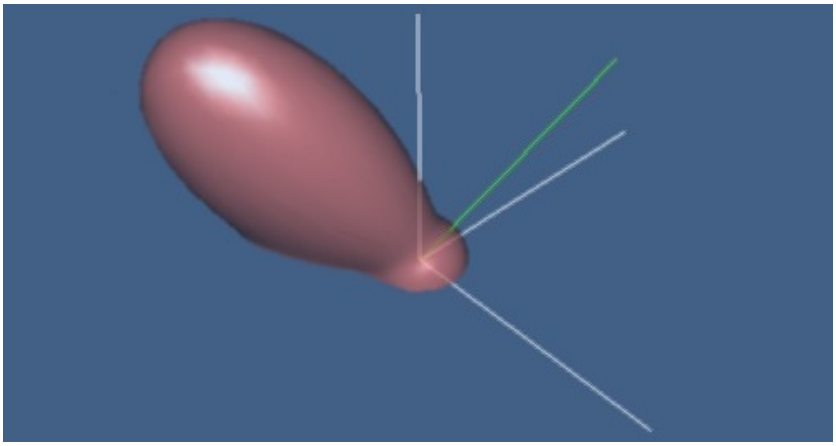


Figure 10.2: The Blinn-Phong BRDF.

Table 10.3: Blinn-Phong BRDF.

Test (s=30)	Target	Unrelaxed	Relaxed
Error between f_r and the rejection sampling BRDF	0.00%	min=4.36% avrg=5.85% max=6.90%	min=0.956% avrg=1.36% max=1.84%
Error between f_r and the direct scatter BRDF	0.00%	min=6.13% avrg=8.3% max=14.9%	min=2.00% avrg=7.6% max=12.2%
BRDF symmetricity error / fraction of energy lost	0.00	min=0.00 avrg=0.00 max=0.00	min=0.00 avrg=0.00 max=0.00
Energy conservation	≤ 1.00	min=0.903 avrg=0.981 max=1.001	min=0.954 avrg=0.984 max=1.00

10.4 The Phong BRDF

The Phong [11] BRDF verified for physical plausibility is:

$$f_r(x, \Psi_i \rightarrow \Psi_o) = \frac{\rho_d}{\pi} + \frac{\rho_s \left(\frac{s+2}{2\pi} \cos^s \phi\right)}{|\cos(N_x, \Psi_i)|} \tag{10.1}$$

This formulation is a generalisation of the specular BRDF by using the same $\cos^s$ random vector PDF that is used by the Blinn-Phong model. The normalisation constant was experimentally chosen such that energy is conserved.

This formulation found to be physical plausible is different from the formulations given in other publications such as by Lewis [62], Lafortune [63], Dutré et al. [3] and Willers [64]. The $s+2$ normalisation (s is the specular exponent) given here ensures energy conservation even for specular exponents smaller than five. Figure 10.3 shows the resulting Phong BRDF. Table 10.4 shows the results of the BRDF analysis.

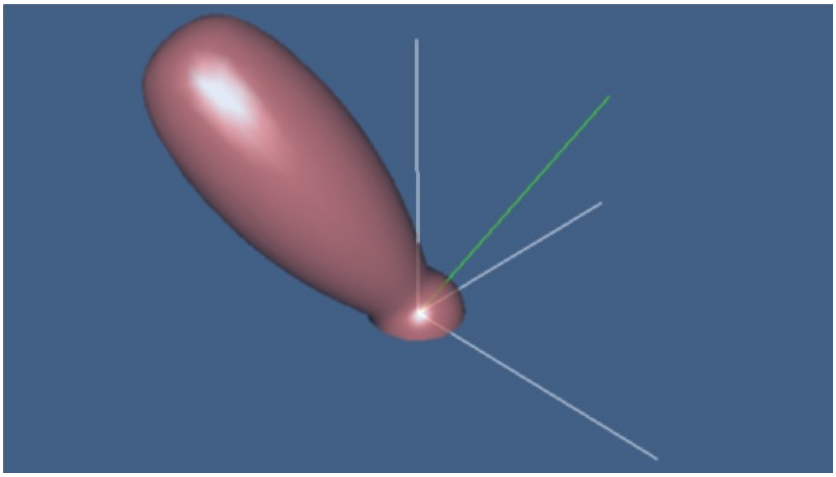


Figure 10.3: The Phong BRDF.

Table 10.4: Phong BRDF.

Test (s=30.0)	Target	Unrelaxed	Relaxed
Error between f_r and the rejection sampling BRDF	0.00%	min=4.65% avrg=6.32% max=8.54%	min=1.05% avrg=1.20% max=1.44%
Error between f_r and the direct scatter BRDF	0.00%	min=4.79% avrg=10.9% max=14.4%	min=2.20% avrg=8.80% max=12.4%
BRDF symmetricity error / fraction of energy lost	0.00	min=0.033 avrg=0.246 max=0.789	min=0.032 avrg=0.238 max=0.782
Energy conservation	≤ 1.00	min=0.942 avrg=1.00 max=1.06	min=0.998 avrg=1.00 max=1.01

10.5 The Glossy BRDF

The glossy BRDF was derived in Chapter 8. It uses a Gaussian scatter lobe:

$$f_r(x, \Psi_i \rightarrow \Psi_o) = \frac{\rho_s \left(\frac{1}{2\pi\sigma^2} \; e^{\frac{-\phi^2}{2\sigma^2}} \right)}{|\cos(N_x, \Psi_i)|} \tag{10.2}$$

Figure 10.4 shows the resulting glossy BRDF. Table 10.5 shows the results of the BRDF analysis.

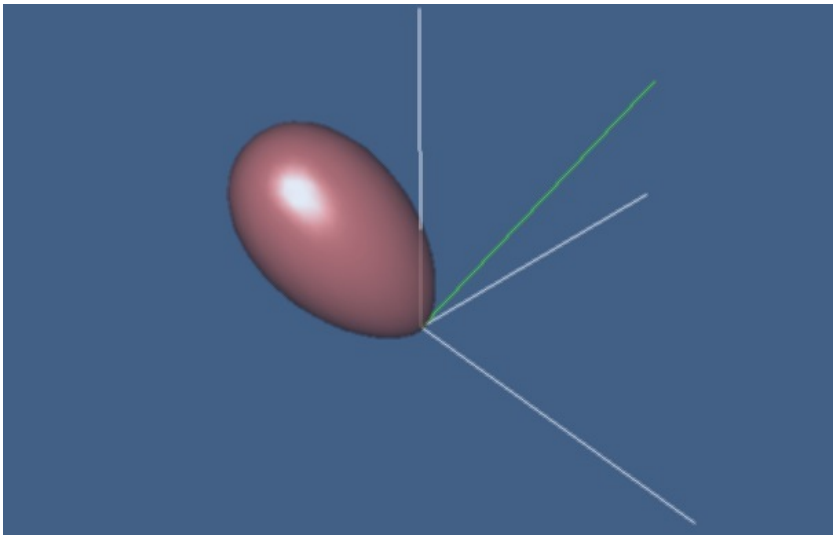


Figure 10.4: The Glossy BRDF.

Table 10.5: Glossy BRDF.

Test (σ=0.1)	Target	Unrelaxed	Relaxed
Error between f_r and the rejection sampling BRDF	0.00%	min=3.94% avrg=5.98% max=9.50%	min=1.79% avrg=2.09% max=2.25%
Error between f_r and the direct scatter BRDF	0.00%	min=6.40% avrg=11.0% max=13.8%	min=2.76% avrg=7.9% max=11.9%
BRDF symmetricity error / fraction of energy lost	0.00	min=0.010 avrg=0.148 max=0.429	min=0.010 avrg=0.144 max=0.431
Energy conservation	$\leq$ 1.00	min=0.864 avrg=0.993 max=1.06	min=0.982 avrg=0.987 max=0.995

10.6 Analysis

The *Error between f_r and the rejection sampled BRDF* results when using the relaxed vector bins are, as expected, quite low. A low error for this test confirms that the BRDF is accurately recovered from a scatter distribution and places confidence in the implementation of the `VerifyBRDFImpl()` test shown in Figure 9.9.

The *Error between f_r and the direct scatter BRDF* results are on average around 8.00% for the relaxed vector bin set. On further inspection it is evident that the error is tied to the symmetricity error for the Phong and Glossy BRDFs. For the Blinn-Phong BRDF the error is likely due to the shadowing and masking effects of the random microfacets not fully taken into account yet in the BRDF implementation. The impact of the masking and shadowing effects of a microfacet surface increases towards grazing (i.e. near horizontal) BRDF angles. Further work is required to incorporate shadowing and masking effects into the other aspects of the Blinn-Phong implementation and possibly the other BRDFs.

The difference between the halfway(H)-vector based BRDFs (i.e. Torrance-Sparrow and Blinn-Phong) and the ϕ-vector based BRDFs (i.e. Phong and the new glossy BRDF) is interesting because the latter is not strictly symmetric, but still often used. Note the presence of the Ψ_o terms in the denominator of the Torrance-Sparrow and Blinn-Phong BRDFs that mimic the Ψ_i terms and bring symmetry to the BRDF.

The *BRDF symmetricity* results do indeed show that the Diffuse BRDF and the Torrance-Sparrow based Blinn-Phong BRDF have, as predicted, zero BRDF symmetricity error while the Phong and Glossy BRDFs potentially have quite large symmetricity errors. The symmetricity error is dependent on the scatter lobe width. The implication of BRDF symmetricity error is a discrepancy between the importance assigned to a transport path and the energy transported along the path.

The *Energy conservation* results show that the adapted BRDFs are energy conserving. The average energy conservation results are fairly robust against unrelaxed vector bin sets because the tension in the vector bins average out

Table 10.6: Dutré et al.'s BRDFs.

Material	$f_r(x, \Psi_i \rightarrow \Psi_o)$
Lambertian Diffuse	$\frac{\rho_d}{\pi}$
Mirror-like Specular	$\frac{\rho_s \delta(\phi)}{\lvert\cos(N_x, \Psi_i)\rvert}$
Phong	$\frac{\rho_d}{\pi} + \frac{\rho_s \cos^s \alpha}{\lvert\cos(N_x, \Psi_i)\rvert}$
Blinn-Phong	$\frac{\rho_d}{\pi} + \rho_s \lvert\cos^s(N_x, \Psi_H)\rvert$
Torrance-Sparrow	$\frac{\rho_d}{\pi} + \frac{\rho_s D(\Psi_H)}{\pi \lvert\cos(N_x, \Psi_i)\rvert \lvert\cos(N_x, \Psi_o)\rvert}$

over the sphere of vectors. Note, however, the discrepancies between the min and max conservation values over the test domain. The differences are significantly higher for the unrelaxed vector bin set.

The differences in the BRDF formulations provided by the various authors are significant. The book by Dutré et al. [3] contains BRDF formulas for, among others: diffuse; specular; Phong; Blinn-Phong; and general Torrance-Sparrow BRDFs. The formulas they give are shown in Table 10.6.

Willers [64] shows formulations of diffuse, specular, Phong and what is effectively the general Torrance-Sparrow BRDFs. The formulas he gives are shown in Table 10.7.

The book by Pharr and Humphreys [10] on the Physically Based Ray Tracing (PBRT) system contains BRDF formulas for, among others, physically plausible diffuse, specular, microfacet based Blinn-Phong and general Torrance-Sparrow models. The formulas they give are shown in Table 10.8.

Dutré et al. and Willers have a π in the denominator of the Torrance-Sparrow formulation while Pharr and Humphreys have a 4. Pharr and Humphreys do not explicitly show a Phong BRDF, but the formulations given by Willers and Dutré et al. differ quite a bit in the normalisation of the specular component. Although, verified physical plausible BRDF *imple-*

Table 10.7: Willers' BRDFs.

Material	$f_r(x, \Psi_i \rightarrow \Psi_o)$
Lambertian Diffuse	$\frac{\rho_d}{\pi}$
Mirror-like Specular	$\frac{\rho_s \delta(\phi)}{\lvert\cos(N_x, \Psi_i)\rvert}$
Phong	$\frac{\rho_d}{\pi} + \frac{\rho_s \left(\frac{s+1}{2\pi} \cos^s \alpha\right)}{\lvert\cos(N_x, \Psi_i)\rvert}$
Blinn-Phong	—
Torrance-Sparrow	$\frac{\rho_d}{\pi} + \frac{\rho_s D(\Psi_H)}{\pi \lvert\cos(N_x, \Psi_i)\rvert \lvert\cos(N_x, \Psi_o)\rvert}$

Table 10.8: Pharr and Humphreys' BRDFs.

Material	$f_r(x, \Psi_i \rightarrow \Psi_o)$
Lambertian Diffuse	$\frac{\rho_d}{\pi}$
Mirror-like Specular	$\frac{\rho_s \delta(\phi)}{\lvert\cos(N_x, \Psi_i)\rvert}$
Phong	—
Blinn-Phong	$\frac{\rho_d}{\pi} + \frac{\rho_s \left(\frac{s+2}{2\pi} \lvert\cos^s(N_x, \Psi_H)\rvert\right)}{4 \lvert\cos(N_x, \Psi_i)\rvert \lvert\cos(N_x, \Psi_o)\rvert}$
Torrance-Sparrow	$\frac{\rho_d}{\pi} + \frac{\rho_s D(\Psi_H)}{4 \lvert\cos(N_x, \Psi_i)\rvert \lvert\cos(N_x, \Psi_o)\rvert}$

mentations is a critical requirement for implementing and comparing rendering algorithms, the differences between BRDF formulations presented in the literature cause confusion.

The BRDF formulations given by Pharr and Humphreys [10] match the formulations adapted to be physically plausible at the hand of the numerical verification. Pharr and Humphreys also provide detailed analytical derivations of their physical plausible BRDFs which instils further confidence in the numerical verification presented here.

Chapter 11

Summary

This chapter summarises the development of the glossy BRDF for LBT. The glossy BRDF equation derived for use in the LBT rendering algorithm is Equation 8.4:

$$f_r(y, \Psi', \vec{\phi}) = \frac{\rho_s e^{-\frac{\phi^2}{2\sigma^2}}}{2\pi\sigma^2 |\cos(N_y, \Psi')|}$$

This glossy BRDF implementation has been verified to be physically plausible. It resembles the specular component of the original Phong [11] BRDF and the modified Phong BRDF used by Lafortune and Williams [57] for physically based rendering.

However, the Glossy BRDF uses a spherical Gaussian PDF for the scattered vectors instead of the cosine PDF used in the Phong BRDF. The Gaussian is more efficient to evaluate and the convolution of two spherical Gaussians is a spherical Gaussian—a property that is used later in Part IV of the thesis. The symmetric scatter lobe, as opposed to the Torrance-Sparrow half angle parameterisation, results in some symmetricity error away from the specular scatter direction for wide lobes. This is, however, not a problem, given the lobe sizes for glossy LBT.

Wang et al. [50] and Xu et al. [52], among others, also discuss the use of spherical Gaussian functions for representing BRDFs. However, their BRDFs exclude the normalisation term and are, as a consequence, not physically

Table 11.1: Summary of physically plausible BRDFs.

Material	$f_r(x, \Psi_i \rightarrow \Psi_o)$
Lambertian Diffuse	$\frac{\rho_d}{\pi}$
Mirror-like Specular Mirror	$\frac{\rho_s \delta(\phi)}{\lvert\cos(N_x, \Psi_i)\rvert}$
Phong	$\frac{\rho_d}{\pi} + \frac{\rho_s\left(\frac{s+2}{2\pi}\cos^s\phi\right)}{\lvert\cos(N_x, \Psi_i)\rvert}$
Torrance-Sparrow	$\frac{\rho_d}{\pi} + \frac{\rho_s D(\Psi_H)}{4\lvert\cos(N_x, \Psi_i)\rvert\lvert\cos(N_x, \Psi_o)\rvert}$
Blinn-Phong	$\frac{\rho_d}{\pi} + \frac{\rho_s\left(\frac{s+2}{2\pi}\lvert\cos^s(N_x, \Psi_H)\rvert\right)}{4\lvert\cos(N_x, \Psi_i)\rvert\lvert\cos(N_x, \Psi_o)\rvert}$
SG Glossy	$\frac{\rho_s\left(\frac{1}{2\pi\sigma^2}\ e^{\frac{-\phi^2}{2\sigma^2}}\right)}{\lvert\cos(N_x, \Psi_i)\rvert}$

plausible.

A process for numerically verifying the implementation of the glossy BRDF and other future BRDFs has been put in place. Recall that the BRDF equations that were found not to be physically plausible were modified as required. The modifications involved visualisation and metering of the binned test results combined with some experimentation. As far as I am aware, numerical verification of the physical plausibility of BRDF implementations has not previously been shown in any published works.

Such a verification step is crucial in any modelling and simulation effort. The formulations given by Pharr and Humphreys [10] match the formulations I found to be physically plausible. Pharr and Humphreys also provide detailed analytical derivations of the BRDF expressions in their book which take into account the conditions for physical plausibility. The correlation between their formulations and mine therefore instils further confidence in the numerical verification presented here. Table 11.1 gives the physically plausible expressions for some well known BRDFs. It would be exciting to, in future, include anisotropic BRDFs such as Ward's [48] BRDF.

Part IV

The Glossy Light Beam Tracing Algorithm

Based on the positive results of Part II and Part III this part formally derives the proposed irradiance estimate and further develops the single- and multi-bounce glossy LBT algorithms to render $L(S|G)^*D$ transport paths. Most of the information was originally published in a research paper [65].

Chapter 12

Derivation of The Glossy Irradiance Estimate

This chapter gives the derivation of the glossy irradiance estimate. The argument follows from the conjecture presented in Part II and the glossy BRDF derived in Part III.

12.1 Overview

In this chapter classical specular LBT as presented in Chapter 4 is used as a basis for single bounce *glossy* LBT. The next chapter, Chapter 13, extends LBT to include multiple bounces, using a light image similar to the one used by Brière and Poulin [27].

LBT proceeds as outlined in Algorithm 2. The light phase is implemented in `TraceBeams` and the forward phase using the `Gather` procedure. As in Chapter 4 the `Render` procedure calls `TraceBeams` and does the parallel for-loop over the pixels in the image to gather the radiance from the scene.

This algorithm is very similar to classical LBT shown in Algorithm 1. The modifications required for glossy beams are on line 13, line 19 and line 21 of Algorithm 2. The `getContributingBeamSegments` method that queries the light beam BVH is updated to return the beams that *potentially* contribute irradiance to the intersection point based on their *glossy* beam bounding

Algorithm 2 Light beam tracing.

```
1: procedure RENDER                                   ▷ Renders one image frame.
2:     TRACEBEAMS
3:     for all pixel ∈ image do                                   ▷ in parallel
4:         L_pixel = GATHER(CameraRay(pixel))
5:     end for
6: end procedure

7: procedure TRACEBEAMS
8:     ...                                              ▷ Trace light beams.
9:     ...                                                ▷ See Section 4.2.
10: end procedure

11: procedure GATHER(ray)
12:     intersect = scene.calcIntersection(ray)

13:     bList=BVH.getContributingBeamSegments(intersect)

14:     for all b ∈ bList do
15:         Area_⊥ = b.calcOrthArea(intersect)
16:         Ψ̄ = b.calcDirection(intersect)
17:         N_i = intersect.normal
18:         Φ = b.flux = ρ_s Φ_s
19:         P_f = b.calcFluxProb(intersect)
20:
21:         ray.L += (ρ_d/π) (|Ψ̄·N_i| / Area_⊥) × Φ × P_f          ▷ See Equation 12.6
22:     end for

23:     ray.L += intersect.L_e

24:     GATHER(intersect.specReflRay)              ▷ Whitted forward render...
25:     ray.L += ρ_s · intersect.specReflRay.L
26: end procedure
```

volumes.

As in Algorithm 1, line 21 gathers together the irradiance contribution from each beam. The expression used in Algorithm 1 to compute that irradiance contribution is replaced with an expression that accounts for glossy surface interaction when computing irradiance. This expression corresponds to the right-hand side of Equation 12.6, an equation that is derived in the next section. P_f is the volume under the spherical Gaussian PDF which represents the fraction of the beam flux that contributes irradiance to x.

12.2 Derivation

This section contains an argument for the conjecture that the irradiance due to a single bounce glossy beam may be calculated using Equation 5.1. The equation is repeated below for easy reference:

$$E(x) = \frac{\Phi_s}{A_\perp} |\cos(N_x, \overline{\Psi})| \iint_\Omega \rho(\vec{\phi}) d\omega_\phi. \qquad (5.1)$$

Figure 5.3 is repeated here as Figure 12.1. It shows an image of a single bounce light beam and I derive the formula for the irradiance $E(x)$ incident at x due to the light beam reflected from the free surface at y. The radiance perceived by the eye is independent of the direction vector Θ from x to the *eye*, but Θ is included for context.

The differential *radiance* incident at x *from* direction Ψ, $dL(x \leftarrow \Psi)$, may be expressed in terms of the BRDF, f_r:

$$\begin{aligned} dL(x \leftarrow \Psi) &= dL(y \rightarrow -\Psi) \\ &= f_r(y, \Psi' \leftrightarrow -\Psi) L(y \leftarrow \Psi') d\omega_{\Psi'} |\cos(N_y, \Psi')| \\ &= f_r(y, \Psi' \leftrightarrow -\Psi) dE_\perp(y \leftarrow \Psi') |\cos(N_y, \Psi')|. \end{aligned}$$

$dE(y \leftarrow \Psi')$ is the differential irradiance at y due to a differential solid angle $d\omega_{\Psi'}$ around the incoming direction Ψ'. $dE_\perp(y \leftarrow \Psi')$ is the differential

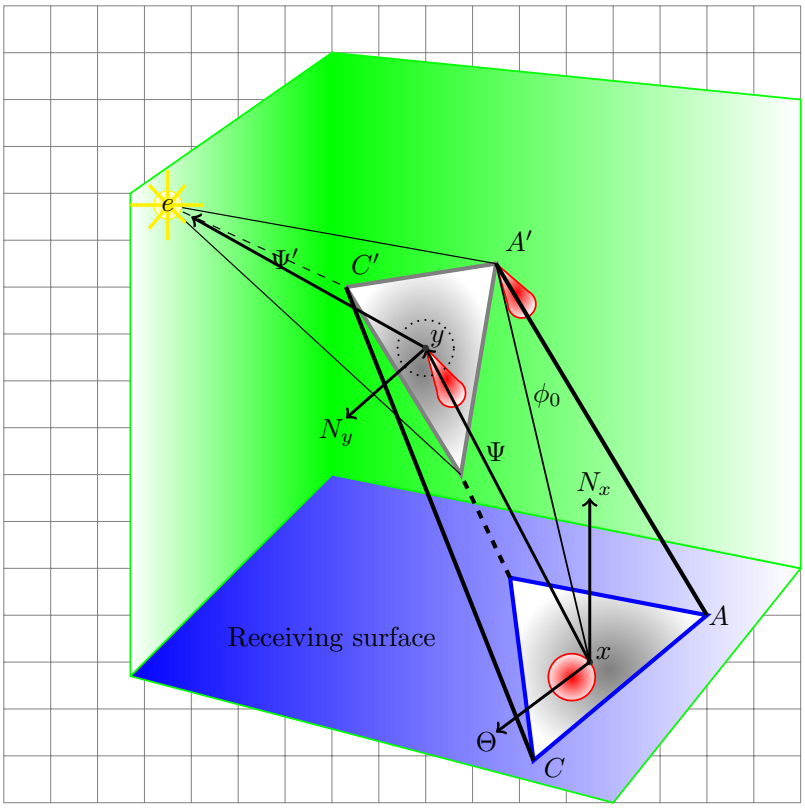


Figure 12.1: The surface at y (a free surface) scatters a light beam onto a diffuse receiving surface at x. The PDF of the glossy scattered vectors is shown in red at y. The PDF of the vectors scattered due to the diffuse receiving surface at x is shown for context.

orthogonal flux density for point y from $d\omega_{\Psi'}$ around direction Ψ'[1].

For a distant, approximately directional, source e and using the definition of the glossy BRDF from Equation 8.3, the *radiance* at x may be expressed as:

$$\begin{aligned} L(x \leftarrow \Psi) &\approx f_r(y, \Psi' \leftrightarrow -\Psi) E_\perp(y) |\cos(N_y, \Psi')| \\ &= \left[\rho_{sy}\rho_y\left(\vec{\phi}(\Psi)\right)\frac{1}{|\cos(N_y, \Psi')|}\right] E_\perp(y) |\cos(N_y, \Psi')| \\ &= \rho_{sy}\rho_y\left(\vec{\phi}(\Psi)\right)\frac{\Phi_s}{A_{y\perp}}. \end{aligned} \tag{12.1}$$

[1]The arrow again indicates the direction of flow of radiance.

Note that a symmetric BRDF, such as the Torrance-Sparrow BRDF shown in Table 11.1 would also be dependent on the outgoing direction Ψ via for example a $\frac{1}{|\cos(N_y,\Psi)|}$ term. Using a different BRDF could therefore lead to the $L(x \leftarrow \Psi)$ also containing a $\frac{1}{|\cos(N_y,\Psi)|}$ term. Further exploration using other BRDFs is left as future work.

Continuing with the derivation using the glossy BRDF, Equation 12.1, the orthogonal flux density ($E_\perp$) is given by $\frac{\Phi_s}{A_{y\perp}}$. The specular coefficient and scatter vector PDF at y are denoted by ρ_{sy} and ρ_y respectively. Recall that Φ_s is the flux contained within the specular beam and $A_{y\perp}$ is the beam cross section at y. The approximations are due to the assumption that the light source is relatively far away compared to the size of the area $A_{y\perp}$.

Then, using Equation 12.1, the differential irradiance $dE(x \leftarrow \Psi)$ at x from direction Ψ and the resulting irradiance, $E(x)$, becomes:

$$\begin{aligned} dE(x) &= L(x \leftarrow \Psi)\,|\cos(N_x,\Psi)\,|d\omega_\Psi \\ &\approx \rho_{sy}\rho_y\left(\vec{\phi}(\Psi)\right)\frac{\Phi_s}{A_{y\perp}}|\cos(N_x,\Psi)\,|d\omega_\Psi \\ E(x) &\approx \iint_{\Omega_{ABC}} \rho_{sy}\rho_y\left(\vec{\phi}(\Psi)\right)\frac{\Phi_s}{A_{y\perp}}|\cos(N_x,\Psi)\,|d\omega_\Psi. \end{aligned}$$

The cosine function changes relatively slowly over the scattering surface at y and one may for small Ω_{ABC} assume an approximately constant cosine value:

$$E(x) \approx \rho_{sy}\frac{\Phi_s}{A_{y\perp}}|\cos\left(N_x,\overline{\Psi}\right)| \iint_{\Omega_{ABC}} \rho_y\left(\vec{\phi}(\Psi)\right)d\omega_\Psi. \qquad (12.2)$$

Using the substitution $d\omega_\Psi = \left(\frac{|\cos(N_y,-\Psi)|}{r_{xy}^2}\right)dA_y$ the *area* formulation of Equation 12.2 is:

$$\begin{aligned} E(x) \approx\ & \rho_{sy}\frac{\Phi_s}{A_{y\perp}}|\cos(N_x,\overline{\Psi})| \times \\ & \iint_{A_{ABC}} \rho_y(\vec{\phi}(y))\left(\frac{|\cos(N_y,-\Psi)|}{r_{xy}^2}\right)dA_y. \end{aligned} \qquad (12.3)$$

r_{xy} is the distance between x and y in Figure 12.1.

The variable substitution to transform between $\rho_y(\phi(\vec{\Psi}))$, $\rho_y(\vec{\phi}(y))$ and later to $\rho_y(\vec{\phi})$ is dependent on factors such as the light-scatterer-geometry and the curvature of the surface patch at y. However, when the BRDF is perfectly specular then Equation 12.2 must evaluate to the irradiance due to the specular beam flux Φ_S at x when x is within the specular beam, and to zero otherwise. When x is within the specular beam one may apply Equation 4.1:

$$\begin{aligned} E(x) &\approx \rho_{sy}\left[\frac{\Phi_s}{A_{x\perp}}|\cos(N_x,\overline{\Psi})|\right] \\ &= \rho_{sy}\left[\frac{\Phi_s}{A_{y\perp}}|\cos(N_x,\overline{\Psi})|\right]\frac{A_{y\perp}}{A_{x\perp}} \text{ by algebraic manipulation,} \end{aligned}$$

Equating the right hand side of this expression for $E(x)$ with the right hand side of the alternative expression in Equation 12.3 gives:

$$\begin{aligned} \int_{A_{ABC}} \rho_y(\vec{\phi}(y))\left(\frac{|\cos(N_y,-\Psi)|}{r_{xy}^2}\right)dA_y &= \frac{A_{y\perp}}{A_{x\perp}} \\ &= \frac{A_{y\perp}}{A_{x\perp}}\int_{\Omega_{ABC}}\rho_y(\vec{\phi})d\omega_\phi \\ &= \int_{\Omega_{ABC}}\rho_y(\vec{\phi})\frac{A_{y\perp}}{A_{x\perp}}d\omega_\phi. \end{aligned}$$

Note that the second step is justified because $\int_{\Omega_{ABC}}\rho_y(\vec{\phi})d\omega_\phi$ is equal to one when x is inside the specular beam. This leads to:

$$\frac{A_{y\perp}}{A_{x\perp}}d\omega_\phi \approx \left(\frac{|\cos(N_y,-\Psi)|}{r_{xy}^2}\right)dA_y = d\omega_\Psi. \tag{12.4}$$

$\frac{A_{y\perp}}{A_{x\perp}}$ is therefore the required variable substitution conversion between $\rho_y(\vec{\phi})$ and $\rho_y(\vec{\phi}(\Psi))$. This variable substitution allows one to express and solve the rendering equation in the domain of the scatter lobe as opposed to the receiver's hemisphere. At the moment it is assumed that this substitution also holds for glossy beams. Further work is required to confirm this.

From Equation 12.3 and Equation 12.4:

$$E(x) \approx \rho_{sy}\frac{\Phi_s}{A_{x\perp}}|\cos\left(N_x,\overline{\Psi}\right)|\iint_{\Omega_{ABC}}\rho_y\left(\vec{\phi}\right)d\omega_\phi. \tag{12.5}$$

Finally, using Equation 4.2 the view independent diffuse reflected radiance L for each beam may be calculated from the definition of the diffuse BRDF:

$$L(x\rightarrow\Theta) = \frac{\rho_d}{\pi}\left[\rho_{sy}\frac{\Phi_s}{A_{x\perp}}|\cos\left(N_x,\overline{\Psi}\right)|\iint_{\Omega_{ABC}}\rho_y\left(\vec{\phi}\right)d\omega_\phi\right]. \tag{12.6}$$

The radiance is independent of the direction vector Θ from x to the *eye*, but Θ is included for context.

12.3 Using Gauss' Theorem

Previously, in Chapter 5, I used a domain decomposition and approximate table lookups instead of a Monte Carlo (MC) or similar numerical solutions to efficiently solve the irradiance surface integral. However, domain decompositions combined with approximations often give numerical problems that lead to structured noise in the image.

If the scatter vector PDF can be written as the divergence of a 2D vector field $\vec{F}\left(\vec{\phi}\right)$, i.e. $\rho\left(\vec{\phi}\right) = \nabla\cdot\vec{F}\left(\vec{\phi}\right)$ then Gauss' divergence theorem [66] allows the surface integral over Ω_{ABC} to be replaced by a line integral around the boundary (line loop AB, BC, CA) of the domain of the surface integral:

$$\begin{aligned}\iint_{\Omega_{ABC}}\rho\left(\vec{\phi}\right)d\omega_\phi &= \iint_{\Omega_{ABC}}\nabla\cdot\vec{F}\left(\vec{\phi}\right)\,d\omega_\phi \\ &= \oint_{AB,BC,CA}\vec{F}\left(\vec{\phi}\right)\cdot\vec{n}\,ds_\phi.\end{aligned} \tag{12.7}$$

Here $\vec{n}$ is the outward pointing normal to the boundary line and s is the variable of integration along the boundary. AB, BC, CA are without adornment the boundary lines $\vec{\phi_0}\rightarrow\vec{\phi_1}$, $\vec{\phi_1}\rightarrow\vec{\phi_2}$ and $\vec{\phi_2}\rightarrow\vec{\phi_0}$.

One may derive $\vec{F}\left(\vec{\phi}\right)$ from $\rho\left(\vec{\phi}\right)$ using the polar form of $\vec{\phi}$ as opposed to the Cartesian form $\vec{\phi} = a\hat{u} + b\hat{v}$ used in Chapter 5. The polar form used

is $a = \phi\cos(\theta)$ and $b = \phi\sin(\theta)$. The scalar ϕ is therefore still used to mean $\|\vec{\phi}\|$ and in this case $\phi = \sqrt{a^2 + b^2}$. The spherical Gaussian $\rho\left(\vec{\phi}\right)$ is cylindrically symmetric and $\frac{\partial}{\partial\theta}\vec{F}$ is zero leading to:

$$\begin{aligned}
\rho\left(\vec{\phi}\right) = \nabla\cdot\vec{F}\left(\vec{\phi}\right) &= \frac{1}{\phi}\frac{\partial}{\partial\theta}\vec{F} + \frac{1}{\phi}\frac{\partial}{\partial\phi}\phi\vec{F} = \frac{1}{\phi}\frac{\partial}{\partial\phi}\phi\vec{F} \\
\int_0^{\varphi}\rho(\phi)\ 2\pi\phi\ d\phi &= 2\pi|\varphi\vec{F}(\vec{\varphi})| \\
\frac{\int_0^{\varphi}\rho(\phi)\ 2\pi\phi\ d\phi}{2\pi\varphi} &= |\vec{F}(\vec{\varphi})|. \qquad (12.8)
\end{aligned}$$

$\int_0^{\varphi}\rho(\phi)\ 2\pi\phi\ d\phi$ is the polar form definition of the cumulative density function (CDF) of $\rho\left(\vec{\phi}\right)$ within radius φ. $\vec{\varphi}$ is used as an additional 2D variable and $\varphi = \|\vec{\varphi}\|$. This leads to:

$$\begin{aligned}
|\vec{F}(\vec{\varphi})| &= \frac{CDF(\vec{\varphi})}{2\pi\varphi} \\
\vec{F}(\vec{\varphi}) &= \frac{CDF(\vec{\varphi})}{2\pi\|\vec{\varphi}\|}\frac{\vec{\varphi}}{\|\vec{\varphi}\|} \quad or \quad \frac{CDF(\vec{\varphi})}{2\pi\|\vec{\varphi}\|}\hat{\varphi}.
\end{aligned}$$

For an efficient analytical expression of the CDF, the Gaussian PDF (ρ, shown in Equation 12.9) is first approximated by a raised cosine distribution [67]:

$$\rho(\phi) = \frac{1}{2\pi}e^{-\frac{1}{2}\phi^2} \qquad (12.9)$$

$$\approx \begin{cases} \frac{1}{4\pi}\left(1+\cos\left(\frac{\pi}{2.5}\phi\right)\right) & \text{for } \phi < 2.5 \\ 0 & \text{otherwise} \end{cases} \qquad (12.10)$$

From $CDF(\vec{\varphi}) = \int_0^{\varphi} \rho(\phi)\ 2\pi\phi\ d\omega_\phi$ and Equation 12.10:

$$
\begin{aligned}
CDF(\vec{\varphi}) &\approx \begin{cases} CDF'(\vec{\varphi}) & \text{for } \varphi < 2.5 \\ 1 & \text{otherwise} \end{cases} \qquad (12.11)\\
CDF'(\vec{\varphi}) &\approx \int_0^{\varphi} \rho(\phi)\ 2\pi\phi\ d\omega_\phi \\
&= \int_0^{\varphi} \frac{1}{4\pi}\left(1+\cos\left(\frac{\pi}{2.5}\phi\right)\right)\ 2\pi\phi\ d\omega_\phi \\
&= \int_0^{\varphi} \frac{\phi}{2}\ d\omega_\phi + \frac{1}{2}\int_0^{\varphi} \phi\cos\left(\frac{\pi}{2.5}\phi\right)\ d\omega_\phi \quad \text{See footnote}^2. \\
&= \left[\frac{\phi^2}{4} + \frac{2.5^2}{2\pi^2}\cos\left(\frac{\pi}{2.5}\phi\right) + \frac{2.5}{2\pi}\phi\sin\left(\frac{\pi}{2.5}\phi\right)\right]_0^{\varphi} \\
&= \frac{\varphi^2}{4} + \frac{2.5^2}{2\pi^2}\cos\left(\frac{\pi}{2.5}\varphi\right) + \frac{2.5}{2\pi}\varphi\sin\left(\frac{\pi}{2.5}\varphi\right) - \frac{2.5^2}{2\pi^2}.
\end{aligned}
$$

Then, one may compute the surface irradiance due to a beam using the *boundary line* integral:

$$E(x) \approx \rho_s \frac{\Phi_s}{A_{x\perp}} |\cos(N_x, \overline{\Psi})| \oint_{AB,BC,CA} \vec{F}\cdot\vec{n}\ ds_\phi. \qquad (12.12)$$

It turns out that the integrand $\vec{F}\cdot\vec{n}$ of Equation 12.12 is quite smooth. Even near the origin it is well behaved as shown in Figure 12.2. The integrand is a dot product of $\vec{n}$ which is orthogonal to the boundary line. The shape of the surface plot is therefore dependent on the orientation of the boundary line and the figure shows the integrand for boundary lines parallel to the dashed arrows.

One may use a 3D table lookup to find the value of the integral for each boundary line. The integral decomposition is based on the boundary edges and is independent of the position of x. This could reduce the impact of a lookup accuracy compared to the surface integral lookups in Chapter 5. However, the investigation of this table lookup solution is left as future work.

[2] Use rule 103, $\int x\cos ax\ dx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax + C$, from `http://integral-table.com`, accessed July, 2014.

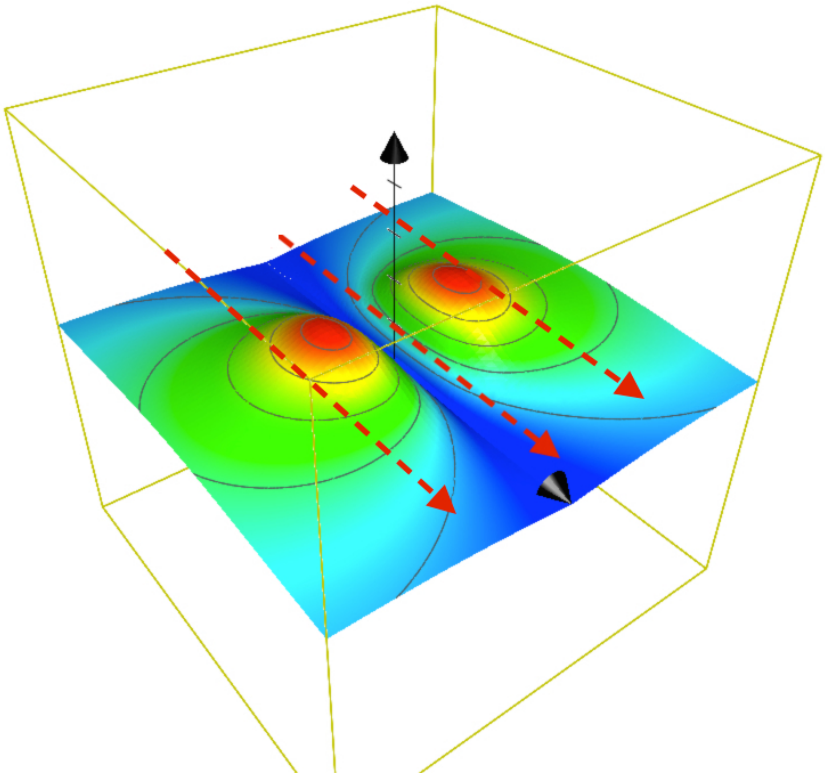


Figure 12.2: A surface plot of the integrand $\vec{F} \cdot \vec{n}$ of Equation 12.12. The figure shows the integrand for boundary lines parallel to the dashed arrows viz. the dashed arrows are orthogonal to $\vec{n}$.

The numerical evaluation of the integral proceeds by first finding the peak of the function along the boundary. The peak is also always the point closest to the origin. Then from this location trapezoidal integration is used in both directions along the border (along a dashed arrow in Figure 12.2) at a dynamic step size of $\frac{1}{2} + \frac{1}{4}\phi$. The step size was experimentally chosen.

Figure 12.3 shows the results of an error analysis of the Gauss solution and the table lookup solution to calculate the probability volumes. The analysis was done by querying the probability volume over a number of randomly placed equilateral triangles. Note that the error values shown are simply the difference between the approximation and the reference probability curve P_{ref}. The source of what looks like noise in `ErrTable` is the difference between the reference solution and the result from the finite resolution table

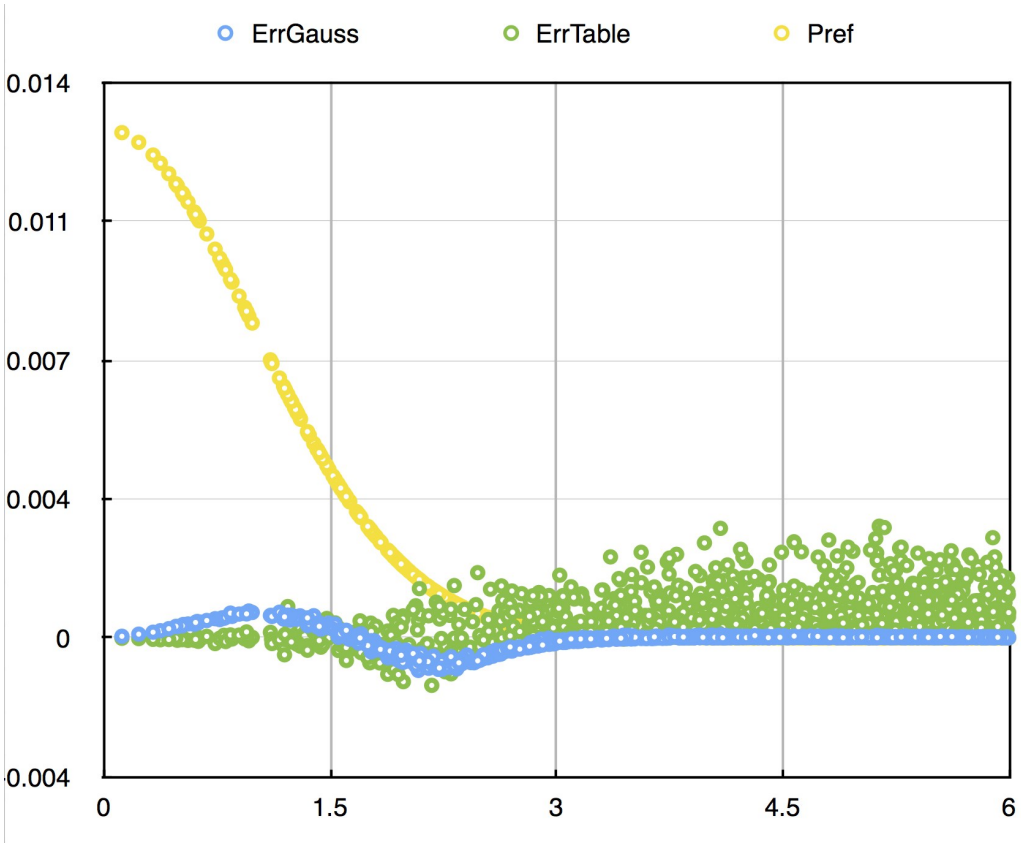


Figure 12.3: The *error* in probability of the table lookup (ErrTable) vs. the error of the Gauss solution (ErrGauss) against the average ϕ of random probability queries. P_{ref} is the reference probability curve.

lookup for each random test case.

As mentioned, the vector function to be integrated in Equation 12.12 makes use of a raised cosine approximation to the Gaussian PDF to allow the CDF to be expressed analytically. The approximation error is evident from the small sinc in the Gauss error plot in Figure 12.3. The error of the Gauss solution is however quite low compared to that of the previous table lookup solution also shown in Figure 12.3.

The execution performance of the new Gauss solution is comparable with the table lookup solution which does three potentially cache unfriendly lookups plus an addition. On a 2.2 GHz Core 2 Duo processor the Gauss solution and the table lookup solution take on average $0.72\mu s$ and $0.75\mu s$ re-

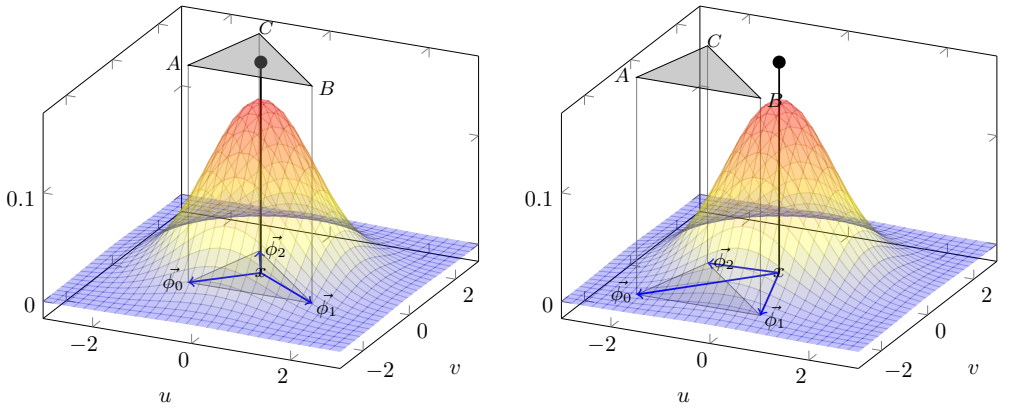


Figure 12.4: Figure on the left shows x within virtual specular beam. Figure on the right shows x outside of the specular beam, but within the glossy beam.

spectively. A MC numerical surface integral takes on average $1.1ms$ and has the disadvantage of increased variance as the surface becomes more specular.

12.4 Shadowing

The occlusion and shadowing of specular light beams are handled by the light image beam refinement. For glossy LBT this equates to occlusions and shadowing already being handled within the virtual specular beam, but not within the glossy region outside of the specular beam. Figure 12.4 shows the two cases. The plot on the left shows x within the virtual specular beam while the plot on the right shows x outside of the specular beam, but within the glossy beam.

Shadow rays therefore only need to be traced when x is outside of the virtual specular beam. Further, the lighting integral domains for irradiance contributions outside of the specular beam never include the peak of the Gaussian probability distribution. The peak of the distribution is only included for points within the specular beam.

When tracing shadow rays one need not trace shadow rays to parts of the scattering surface that have a relatively low contribution to the irradiance. One may therefore conveniently make use the Gaussian distribution to do

importance sampling of the scattering surface when tracing the shadow rays.

Fortunately, the beam refinement is usually at a finer subdivision level near the edges of scattering surfaces. This results in a natural increase of occlusion calculations which seems to handle the shadowing adequately.

Initial tests showed little visual impact when tracing the additional shadow rays. The further study of the impact and benefit of tracing such additional shadow rays is left as future work.

12.5 All-Frequency Surface Interactions

The assumptions that lead to Equation 12.12 are regarding the approximately locally constant $\cos(N_y, \Psi')$ and $\cos(N_x, \Psi)$, the use of the glossy BRDF which is only symmetric for relatively glossy materials as well as the assumption that Equation 12.4 also holds for glossy beams. If one were to incorporate a different BRDF and confirm Equation 12.4 it would be possible to apply Equation 12.12 to all-frequency interactions given the correct domain of integration relative to the axis of an appropriate scatter distribution.

To demonstrate this consider a potential scatter distribution of a diffuse interaction at point y to be $\rho_d(\phi) \propto \cos(\phi)$ for ϕ the angle between the *normal* N_y and the outgoing BRDF direction Ψ. A Gaussian scatter distribution with a wide standard deviation of $\frac{\pi}{5}$ is a good approximation of the scatter produced by a diffuse surface.

Figure 12.5 shows some single bounce diffuse LBT results using such a spherical Gaussian scatter distribution centred around the surface normal. The direct illumination component is again removed to highlight the scattered light. Note that removing the direct illumination makes the diffuse reflector on the right appear black in the extended Whitted raytracer.

All-frequency interaction requires the surface radiance distribution to be known. The next section shows how to do this for multi-bounce *glossy* interactions using LBT. Further analysis of diffuse interaction using this irradiance equation and multi-bounce diffuse interactions is left as future work.

One should in fact be able to draw a parallel between Equation 12.12 and an equation known to Lambert in 1760, and possibly derived by him [8]. The

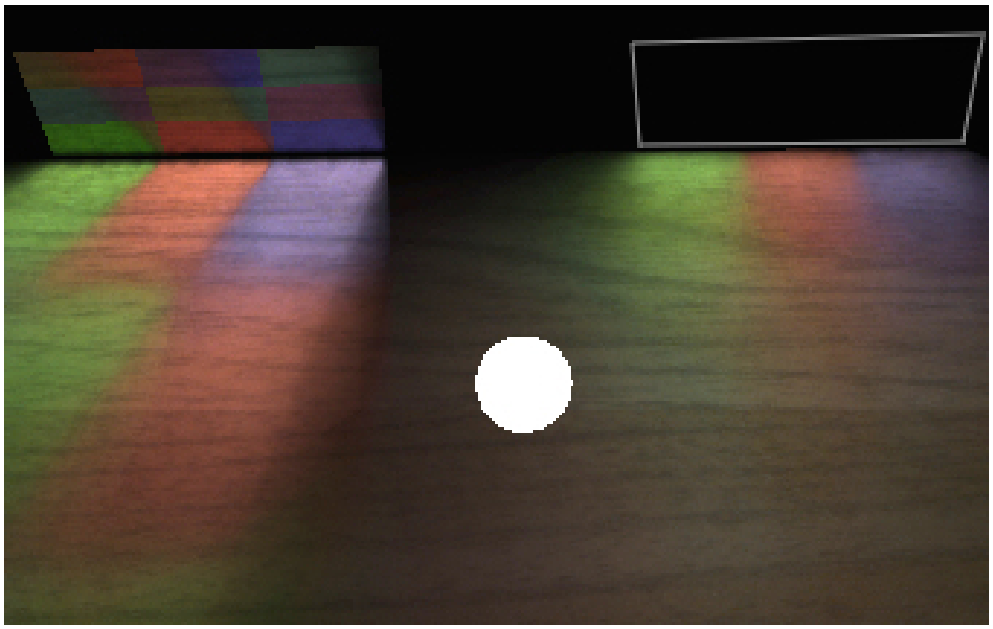


Figure 12.5: Example of single bounce beam tracing with glossy (left) and diffuse (right) scatter distributions.

equation is given by:

$$E(x) = -\frac{L}{2\pi}\sum_{l=1}^{m} B_l \cos\gamma_l \tag{12.13}$$

where

$$\begin{aligned} B_l &= \cos^{-1}\frac{(V_l - x)\cdot(V_{l+1} - x)}{\|V_l - x\|\|V_{l+1} - x\|} \\ \cos\gamma_l &= N_l \cdot N_x \\ N_l &= \frac{(V_l - x)\times(V_{l+1} - x)}{\|(V_l - x)\times(V_{l+1} - x)\|}. \end{aligned}$$

Here V_l is the l'th vertex of the Lambertian source ($l = \{1, 2, 3\}$ shown in Figure 12.6) and $V_{m+1} = V_1$. L is the radiance of the Lambertian source and N_l is the normal to the polygon xV_lV_{l+1}. Note the sign of the equation and that $\cos\gamma_l$ can be negative.

This version of Lambert's equation (adapted from [68] and [69]) for ir-

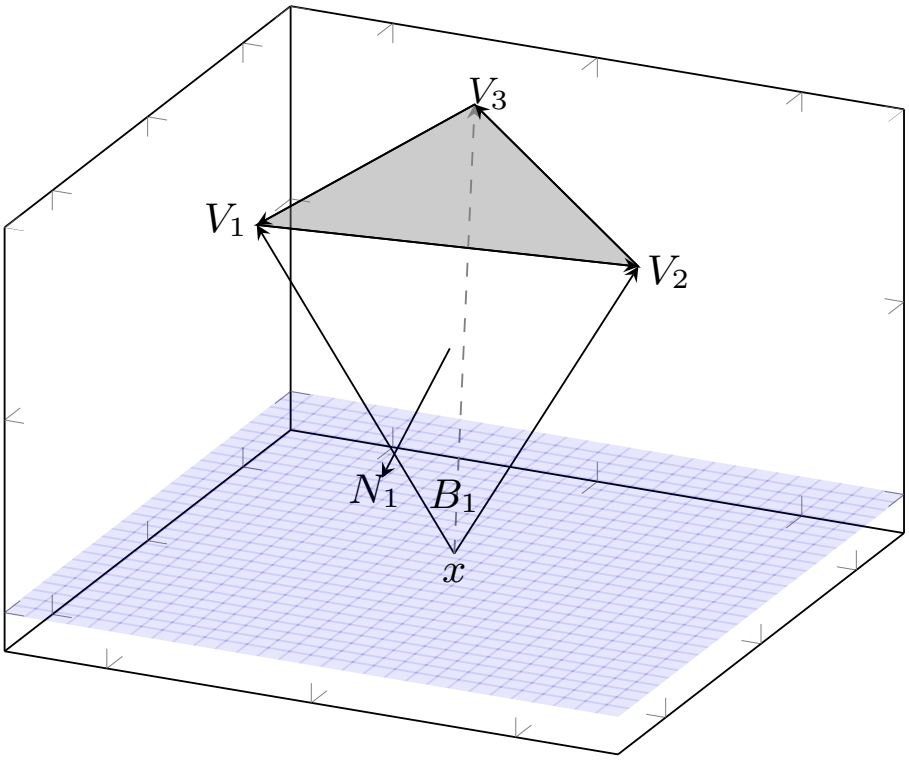


Figure 12.6: Figure accompanying Lambert's equation, Eq 12.13.

radiance due to a Lambertian area source resembles a directed line integral around the boundary of the Lambertian source. Stark and Riesenfeld [69] also show how to use Green's theorem to derive what seems to be an alternative form of Lambert's equation. The two dimensional version of Green's Theorem as used by Stark and Riesenfeld is equivalent to Gauss' theorem in two dimensions. Further investigation of the potential duality between Lambert's equation and Equation 12.12 is also, lamentably, left as future work.

Chapter 13

Illustration of Multi-bounce Glossy Light Beam Tracing

Figure 13.1 shows the value of adding multiple bounces to the rendering of caustics. The same scene and light source are used, but the image on the right shows how light really bounces around inside of such a reflective ring when *not* limiting the simulation to single-bounce transport paths. Figure 13.1 was rendered with the LBT rendering algorithm proposed below.

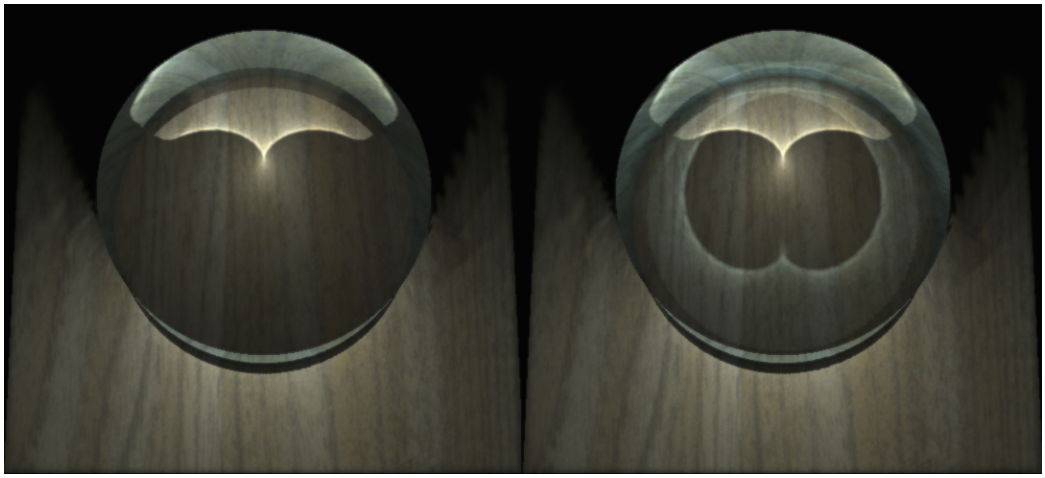


Figure 13.1: Single bounce cardioid caustic (on the left) compared to a more realistic multi-bounce cardioid caustic (on the right).

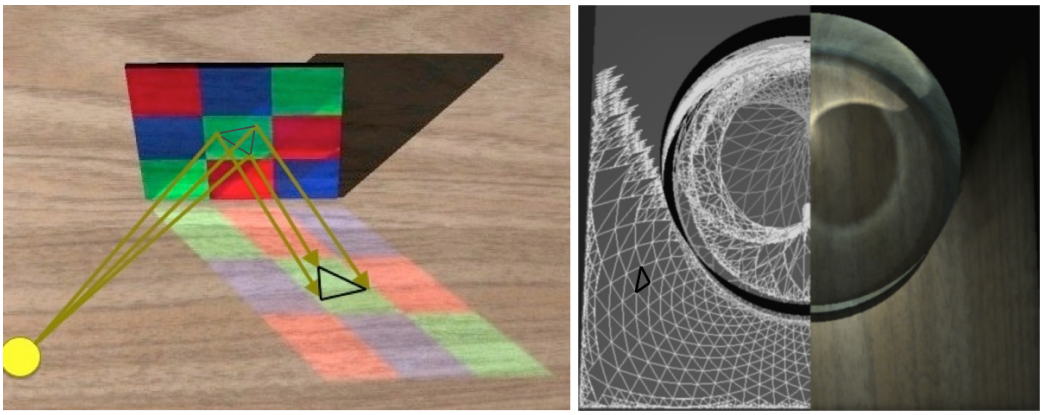


Figure 13.2: Left: Shows a single light beam (bounded by three light paths) reflected off a specular surface. Right: The refined light beams of a cardioid caustic.

13.1 Overview

I build upon the light image idea of Brière and Poulin [27] and the derivation in the previous chapter to show one example of how to extend the single bounce glossy LBT algorithm to *multiple-bounces*.

Brière and Poulin's [27] light image rendering algorithm starts with a low resolution grid that sub-divides a plane placed between the light and the scene. A light beam is then traced through each subdivided element of the plane by tracing light rays along the corners of the grid. Beams with incoherent corner light rays iteratively result in further subdivision of the plane in front of the light source and more beams being traced. This process is continued until the subdivision of the plane appropriately images the geometry of the scene from the light source. At this point the light beams together is the volumetric representation of the specularly scattered light in the scene. Figure 3.5 in the related work chapter shows an example light image taken from Brière and Poulin's paper [27]

Regardless of surface glossiness, I define a specular light path as a virtual light path that interacts with all surfaces as if they are perfectly specular. The truncated specular path between surface interactions is known as a specular light path *segment*. Similarly, a specular light beam is a triplet of specular light paths. The truncated beam between surface interactions is known as a

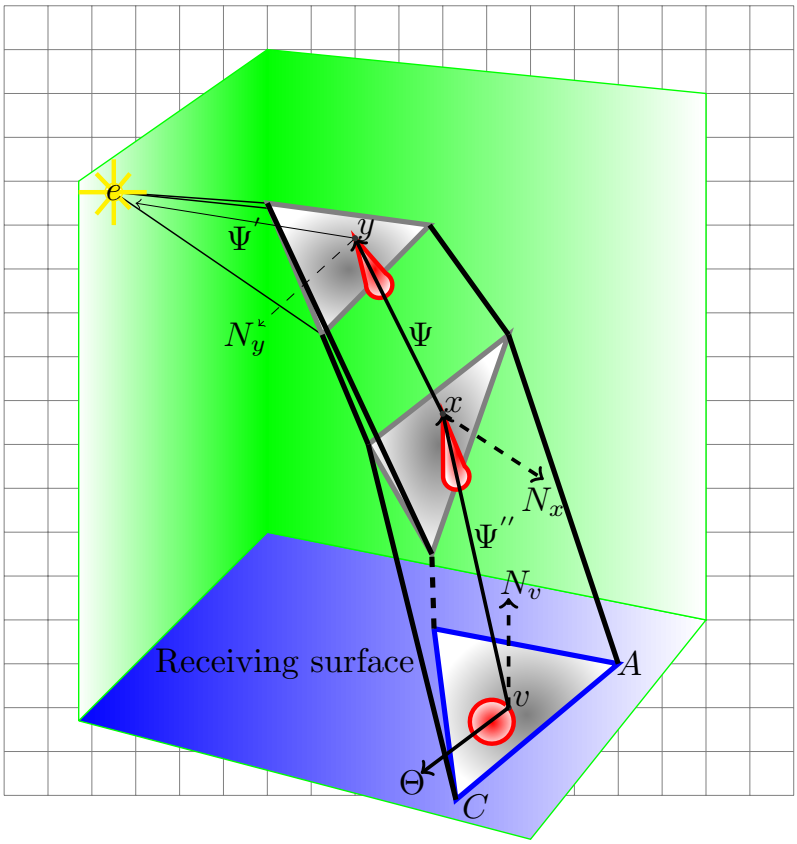


Figure 13.3: The surfaces at x and y are free surfaces. The surface at v is the diffuse receiving surface. The PDFs of the scattered vectors are shown in red.

beam path segment.

Following the lead of Brière and Poulin [27] the sub-division refinement of the light image and the light beam paths are driven by the coherency between neighbouring specular light paths. A light beam path is *coherent* as long as its specular light paths intersect the same object surfaces and stay approximately coaxial. Specularity usually changes slowly over a surface and is assumed here to be constant per object surface.

Figure 13.2 shows what a typical light image mesh projected on the scene looks like for a cardioid caustic. A 2-subdivision where each triangle-side is divided by two is used.

Although not the focus of this thesis, it is worth mentioning that to accelerate the *multi-bounce* beam segment traversal during rendering, a k-sided Discrete Oriented Polytope (k-DOP) [70] BVH was implemented for the purposes of the present study[26][27][54]. A k-DOP is a type of axis aligned bounding box (AABB). A k-DOP BVH was chosen because of the discrete orientations of the bounding planes that seem to create a very tight fitting hierarchy. The k planes (of eack k-DOP node) are directly evaluated during BVH traversal. A 12-DOP seemed to work well for LBT. Using less planes performs significantly worse while more planes seem to offer insignificant benefit. Chapter 16 gives more information on the software and the k-DOP BVH.

The light image subdivision used by Brière and Poulin [27] is a quadtree subdivision of the image. They limit the subdivision level difference between neighbouring quadtree branches to one. Such subdivision allows them to easily fill any cracks between neighbouring beams of different subdivision levels reflected off of curved surfaces. I use a 2-subdivision of triangles as opposed to a 2-subdivision of quads. Note that, as in the case of quads, there could be cracks between neighbouring beams of different subdivision levels reflected off of curved surfaces. The average size of the cracks is, however, kept insignificant by choosing an initial light image mesh resolution that is appropriately high. Fixing or preventing the cracks is left as future challenge for using the light image mesh for multi-bounce LBT.

13.2 Derivation

Figure 13.3 shows the scenario used to derive multi-bounce LBT. An expression for the irradiance $E(v)$ at v is required. The radiance perceived by the eye is independent of the direction vector Θ from x to the *eye*, but Θ is again included for context.

From Figure 13.3:

$$\begin{aligned} dL\left(v \leftarrow -\Psi''\right) &= dL\left(x \rightarrow \Psi''\right) \\ &= f\left(x, \Psi \leftrightarrow \Psi''\right) \\ &\quad L\left(x \leftarrow \Psi\right) |\cos\left(N_x, \Psi\right)| d\omega_\Psi. \end{aligned}$$

Then, using the right hand side of the glossy BRDF definition in Equation 8.3 of Chapter 8 to substitute for $f\left(x, \Psi \leftrightarrow \Psi''\right)$ one gets:

$$dL\left(v \leftarrow -\Psi''\right) = \rho_{sx}\rho_x\left(\vec{\phi_x}\left(\Psi, \Psi''\right)\right) L\left(x \leftarrow \Psi\right) d\omega_\Psi.$$

Using Equation 12.1 to substitute for $L\left(x \leftarrow \Psi\right)$ leads to:

$$\begin{aligned} dL\left(v \leftarrow -\Psi''\right) &\approx \rho_{sx}\rho_x\left(\vec{\phi_x}\left(\Psi, \Psi''\right)\right)\rho_{sy}\rho_y\left(\vec{\phi_y}\left(\Psi', \Psi\right)\right) \times \\ &\quad \frac{\Phi_s}{A_{y\perp}} d\omega_\Psi \end{aligned}$$

and thus

$$\begin{aligned} L\left(v \leftarrow -\Psi''\right) &\approx \rho_{sx}\rho_{sy}\frac{\Phi_s}{A_{y\perp}} \times \\ &\quad \iint_{\Omega_\Psi} \rho_x\left(\vec{\phi_x}\right)\rho_y\left(\vec{\phi_y}\right) d\omega_\Psi. \end{aligned}$$

Note the dependence between $\vec{\phi_x} = \vec{\phi_x}\left(\Psi, \Psi''\right)$ and $\vec{\phi_y} = \vec{\phi_y}\left(\Psi', \Psi\right)$. Both are a function of the vector Ψ. In other words, there is a relationship $\vec{\phi_x} = \vec{\phi} - \vec{\phi_y}$ and $\vec{\phi} = \vec{\phi_x} + \vec{\phi_y}$.

Then to calculate the irradiance:

$$\begin{aligned} dE(v) &= L\left(v \leftarrow -\Psi''\right) |\cos(N_v, \Psi'')| d\omega_{\Psi''} \\ dE(v) &\approx \rho_{sx}\rho_{sy}\frac{\Phi_s}{A_{y\perp}} \times \\ &\quad \iint_{\Omega_\Psi} \rho_x\left(\vec{\phi}_x\right)\rho_y\left(\vec{\phi}_y\right) d\omega_\Psi |\cos(N_v, \Psi'')| d\omega_{\Psi''} \\ E(v) &\approx \rho_{sx}\rho_{sy}\frac{\Phi_s}{A_{y\perp}} |\cos(N_v, \overline{\Psi''})| \times \\ &\quad \iint_{\Omega_{\Psi''}} \iint_{\Omega_\Psi} \rho_x\left(\vec{\phi}_x\right)\rho_y\left(\vec{\phi}_y\right) d\omega_\Psi d\omega_{\Psi''}. \end{aligned} \tag{13.1}$$

As in the derivation of Equation 12.4 we have $d\omega_\Psi = \frac{A_{y\perp}}{A_{x\perp}} d\omega_{\phi_y}$ and $d\omega_{\Psi''} = \frac{A_{x\perp}}{A_{v\perp}} d\omega_{\phi_x}$. Using this as well as Equation 13.1 the irradiance becomes:

$$\begin{aligned} E(v) &\approx \rho_{sx}\rho_{sy}\frac{\Phi_s}{A_{v\perp}} |\cos(N_v, \overline{\Psi''})| \times \\ &\quad \iint_{\Omega_{\phi_x}} \iint_{\Omega_{\phi_y}} \rho_x\left(\vec{\phi}_x\right)\rho_y\left(\vec{\phi}_y\right) d\omega_{\phi_y} d\omega_{\phi_x}. \end{aligned} \tag{13.2}$$

Equation 13.2 expresses the irradiance at v as a function of the *specular* beam flux at v and the angular support of the scattering surfaces at x and y. Given that the domain Ω_{ϕ_y} is large relative to the width of $\rho_y\left(\vec{\phi}_y\right)$ and given the relationship $\vec{\phi}_x = \vec{\phi} - \vec{\phi}_y$ between $\vec{\phi}_x$ and $\vec{\phi}_y$ the double surface integral is a convolution of spherical Gaussians. One may therefore simplify

the expression for $E(v)$:

$$\begin{aligned} E(v) &\approx \rho_{sx}\rho_{sy}\frac{\Phi_s}{A_{v\perp}}|\cos(N_v,\overline{\Psi''})|\times \\ &\iint_{\Omega_{\phi x}}\iint_{\Omega_{\phi y}}\rho_x\left(\vec{\phi_x}\right)\rho_y\left(\vec{\phi_y}\right)d\omega_{\phi_y}d\omega_{\phi_x} \\ &\approx \rho_{sx}\rho_{sy}\frac{\Phi_s}{A_{v\perp}}|\cos(N_v,\overline{\Psi''})|\times \\ &\iint_{\Omega_{\phi-\phi y}}\iint_{\Omega_{\phi y}}\rho_x\left(\vec{\phi}-\vec{\phi_y}\right)\rho_y\left(\vec{\phi_y}\right)d\omega_{\phi_y}d\omega_{\phi} \\ &\approx \rho_s\frac{\Phi_s}{A_{v\perp}}|\cos(N_v,\overline{\Psi''})|\times \\ &\iint_{\Omega_\phi}P\left(\vec{\phi}\right)d\omega_{\phi}. \end{aligned} \tag{13.3}$$

This simplification is valid only when the width of the light beam is wider than the width of the scatter distribution. Otherwise the integral would not describe a convolution. For narrower beams tracing light rays would therefore be more suitable.

I use the uppercase P symbol here instead of the lowercase ρ to indicate the effective or apparent scatter lobe of the beam segment. In the rest of the thesis P is also assumed to be a spherical Gaussian distribution. Now, in order to calculate the irradiance at v one need only know the apparent scatter distribution at the surface at x. The light interactions at earlier scattering surfaces such as y become irrelevant. Therefore, equation 13.3 decouples the beam segments of a light beam path from one another.

For rough glossy and diffuse surfaces the domain Ω_{ϕ_y} might be small relative to the width of $\rho_y\left(\vec{\phi_y}\right)$. Future work should therefore address the accumulation of the scatter lobe in these cases to generalise the rendering algorithm to all-frequency interactions.

The same Gauss' divergence theorem solution is reused for multi-bounce

beam segments, leading to:

$$E(v) \approx \rho_s \frac{\Phi_s}{A_{v\perp}} |\cos\left(N_v, \overline{\Psi''}\right)| \times \oint_{AB,BC,CA} \vec{F} \cdot \vec{n} \; ds_\phi. \tag{13.4}$$

In this case the integration domain is defined by P and Ω_{ABC}.

Chapter 14

Results and Analysis

This chapter presents the results of the proposed single and multi-bounce glossy LBT rendering algorithms. The quality and execution performance of LBT is compared to PM and light tracing.

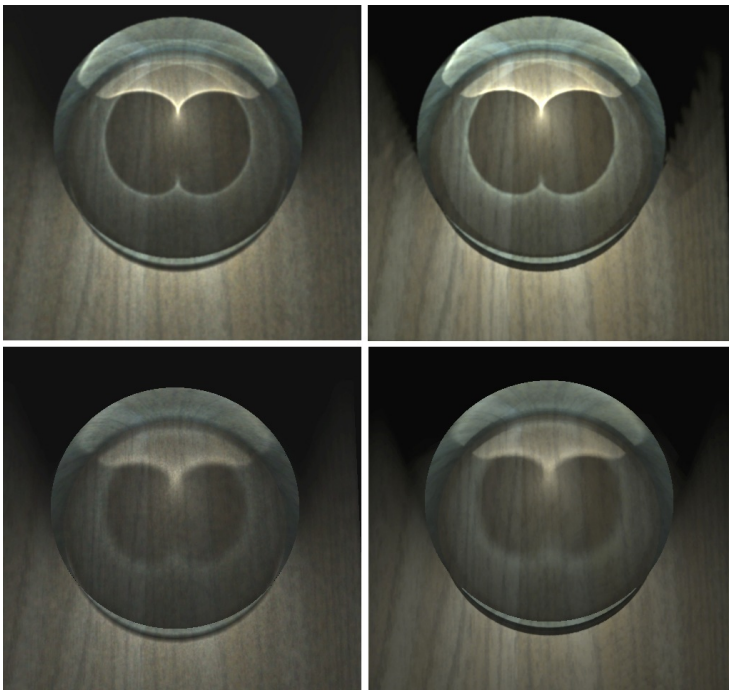


Figure 14.1: Top row compares PM (left) to multi-bounce LBT (right) for $\sigma = 0.001$. Bottom row does the same for $\sigma = 0.05$. 3.3M photons in map vs. 35k light beams.

Table 14.1: Render Times - 3.3M photons & 35k beams

Scenario	Glossy σ	PM(s)	LBT(s)
Core 2:			
Ring	0.001 rad	120 (51+68)	61 (1.7+59)
Ring	0.05 rad	120 (51+68)	130 (1.7+130)
Core i7:			
Ring	0.001 rad	31 (21+10)	10 (1.3+9)
Ring	0.05 rad	31 (21+10)	28 (1.3+27)

14.1 Results

This section presents some performance and comparative quality results for LBT. The two computer platforms were a 2.26 GHz Core 2 Duo Macbook Pro with 8 GB RAM and a quad core 3.2 GHz Core i7 960 with 6 GB RAM. The number of render threads of execution is automatically set to the number of cores reported by C++11's `std::thread`'s `hardware_concurrency()` method. Two are reported in the case of the Core 2 Duo and eight in case of the hyper threaded quad core. All images were rendered at a size of 800x600 pixels. The render times are given as ('light phase time' + 'forward render time') to *two significant digits*.

The LBT results are compared to classical PM using a direct caustic radiance estimate—i.e. no final gather—because such a direct estimate efficiently simulates $L(S|G)^*D$ caustic transport paths. A MC light tracer [15] is used to generate reference results. Due to the noise in the MC light tracer a qualitative comparison of the image quality is done. Future work should investigate a suitable image similarity metric to quantitatively measure the error of the PM and LBT results.

Figure 14.1 compares a simple cardioid caustic result of PM and multi-bounce LBT. The nearest *200* photons are used in the radiance estimate. Notice the softer caustics reflected from the more glossy surfaces in the bottom row. Table 14.1 shows the execution performance of PM and LBT for

Table 14.2: Render Times - 16M photons & 62k beams

Scenario	Glossy σ	PM(s)	LBT(s)
Core 2:			
Gears	0.001 rad	270 (110+160)	130 (3.4+130)
Gears	0.05 rad	270 (110+160)	270 (3.4+270)
Core i7:			
Gears	0.001 rad	54 (31+23)	21 (1.7+19)
Gears	0.05 rad	54 (31+23)	40 (1.7+38)

the cardioid caustic. For this scene the software is capable of tracing 2.4M photons or rays per second on the Core 2 platform and 16M rays per second on the i7 platform. Due to the large number of pixels within an image the render times are, as expected, quite stable over any number of runs.

Figure 14.2 shows the photon map, LBT and reference results of caustics due to more complex gear objects. 200 photons are used in the radiance estimate. Each result is rendered using one rendering algorithm. Table 14.2 shows the execution performance of the three rendering algorithms. For this scene the software is capable of tracing 1.2M photons or rays per second on the Core 2 platform and 7.8M rays per second on the i7 platform.

Note that the reference light trace renderer only has a light phase. Specular and glossy objects therefore appear black as opposed to shiny because light tracing is not suited to rendering the appearance of these objects.

Figure 14.3 shows the reference, photon map and LBT results of a larger scene. 500 photons are used in the radiance estimate. The noise on the walls of the photon map result is due to the same direct radiance estimate and photon map used for all transport paths. Table 14.3 shows the execution performance of the three rendering algorithms for this scene. For this scene the software is capable of tracing 600k photons or rays per second on the Core 2 platform and 3.9M rays per second on the i7 platform.

Table 14.3: Render Times - 53M photons & 185k beams

Scenario	Glossy σ	PM(s)	LBT(s)
Core 2:			
DiscoBall	0.001 rad	960 (300+660)	170 (33+140)
DiscoBall	0.05 rad	960 (300+660)	550 (33+520)
Core i7:			
DiscoBall	0.001 rad	210 (110+95)	34 (14+20)
DiscoBall	0.05 rad	210 (110+95)	88 (14+74)

14.2 Analysis

The results shown all use between 100x and 1000x less beams than photons. However, the LBT results match well with the caustic PM and light tracer results. Due to some remaining noise in the MC light tracer, a qualitative comparison of the image quality is done. Future work should investigate a suitable image similarity metric to quantitatively measure the error of the PM and LBT results.

The LBT rendering algorithm executes as fast or faster than PM for scatter standard deviations up to 0.05 radians. In the case of the more complex disco ball scene that has many glossy surfaces the LBT algorithm is approximately two to six times faster than PM.

The light pass of LBT is also significantly more efficient than the light pass of PM. This is mainly due to the reduced number of light rays that are required.

However, the glossiness (viz. width) of the beams affect the overlap and how many beams potentially contribute irradiance to a surface. This directly impacts the execution speed of the forward rendering phase of LBT.

Regardless of how the beam segments are generated, the average cost of the rendering algorithm for single and multiple bounces stays $\mathcal{O}(\log n)$ in the total number of beam segments. This is due to the traversal of the beam BVH during forward rendering. The forward rendering of both LBT and PM

may be parallelised to the point of having one thread of execution per pixel in the image. The light phase of PM and LBT may also be parallelised, but the photon map and beam BVH should be made thread safe by, for example, atomic operations or making use of a mutex to control access.

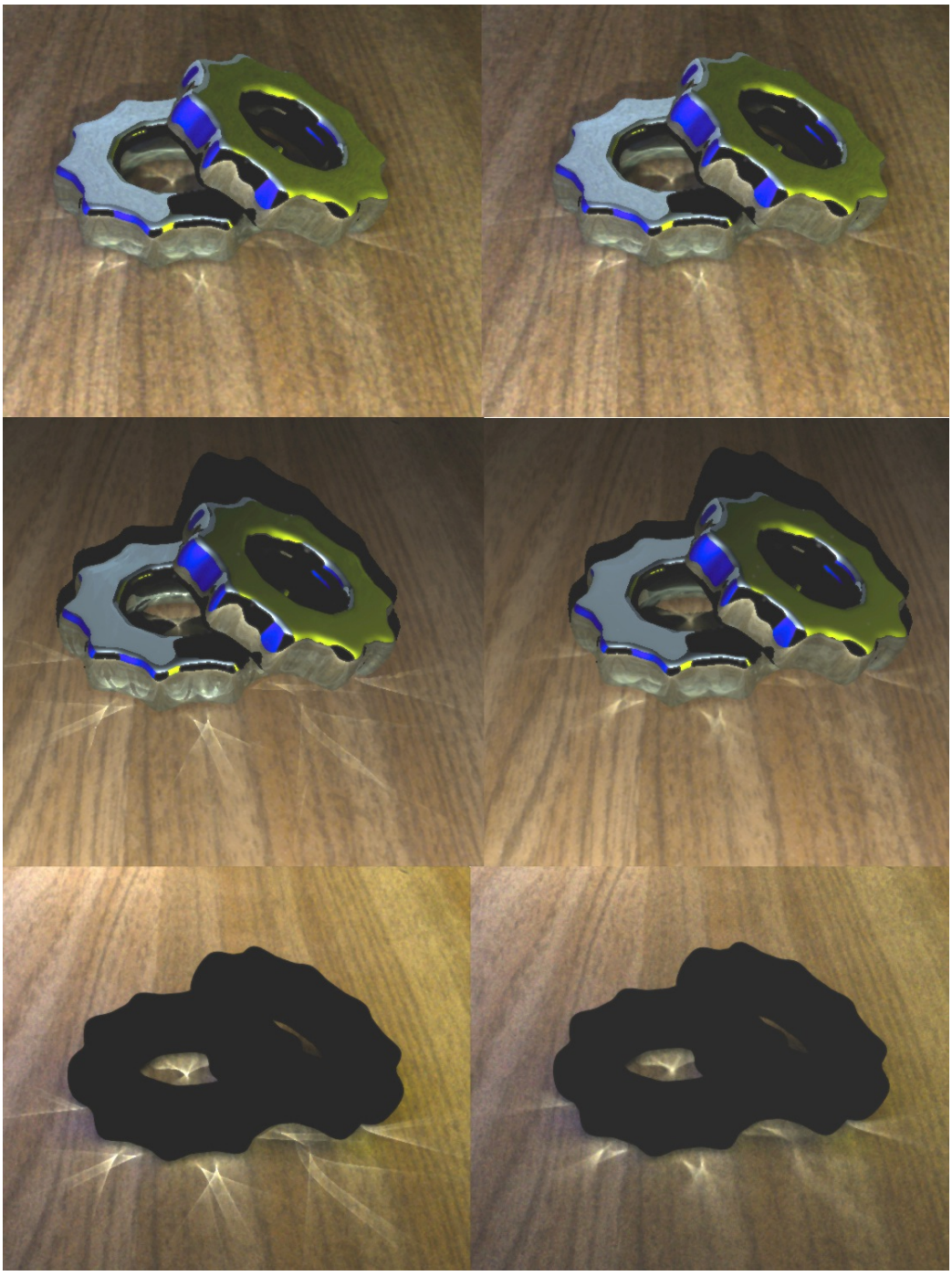


Figure 14.2: The photon map (top), LBT (middle) and MC light trace reference (bottom) results of caustics due to more complex gear objects. The left column shows glossy reflections for $\sigma = 0.001$ radians while the right column shows the same for $\sigma = 0.05$ radians.

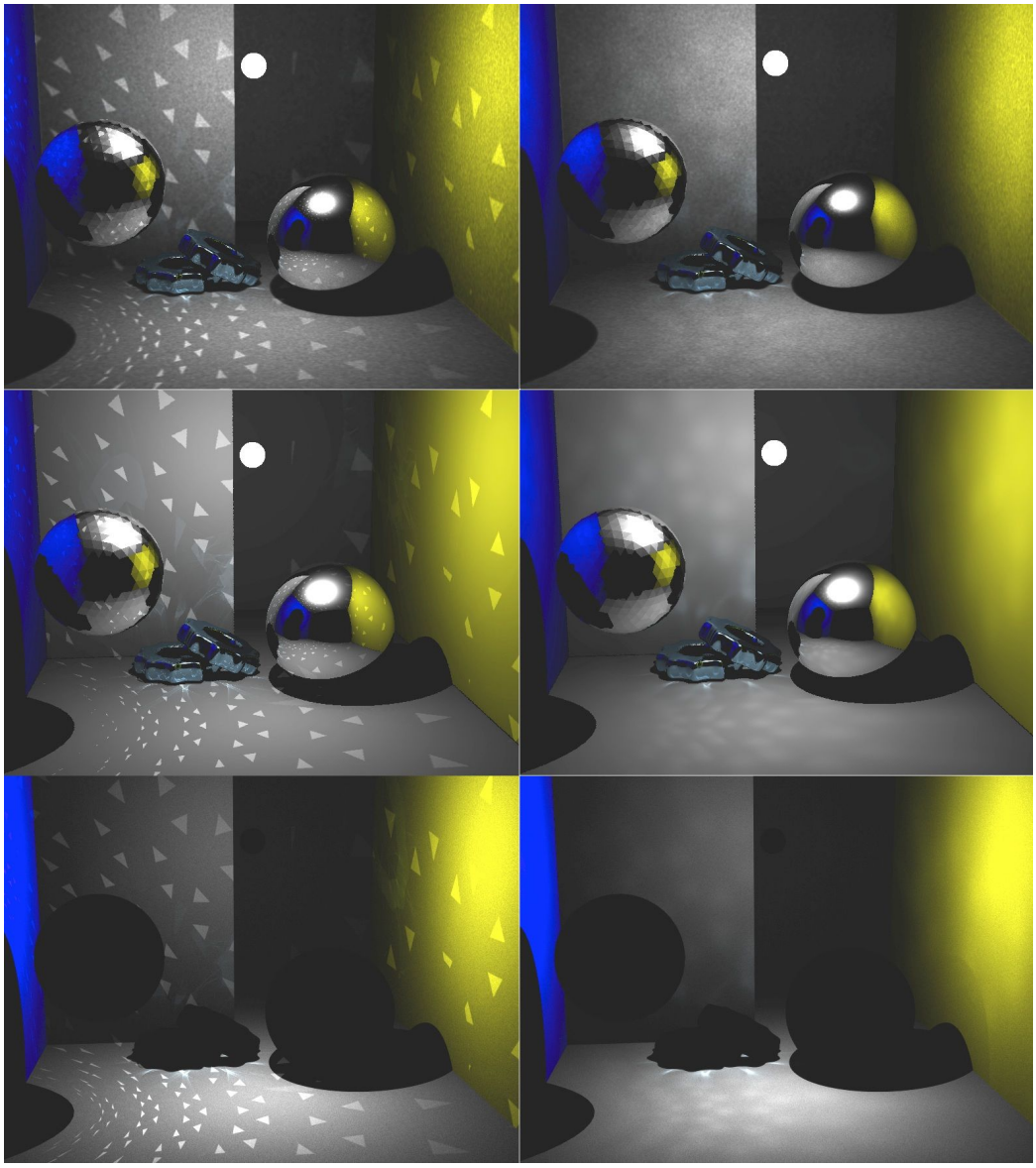


Figure 14.3: The photon map (top), LBT (middle) and MC light trace reference (bottom) results of caustics due to more complex disco ball scene. The left column shows glossy reflections for $\sigma = 0.001$ radians while the right column shows the same for $\sigma = 0.05$ radians.

Chapter 15

Summary

LBT has been extended to include glossy interactions. This chapter summarises the proposed single- and multi-bounce LBT rendering algorithms. The multi-bounce solution relies on the fact that the beam width is relatively large compared to the width of the scatter lobes. However, the average cost of having included multiple bounces stays $\mathcal{O}(\log n)$ in the total number of beam segments.

Gauss' divergence theorem is used to express the beam segment lighting integral as a boundary line integral. However, a raised cosine approximation is required for efficient evaluation of the CDF. This introduces some locally constant error as shown in Figure 12.3, but exhibits much improved noise behaviour over the previous table lookup optimised solution. The error could be reduced further with a more accurate approximation of the Gaussian distribution's CDF. A future topic for research could therefore be to evaluate other approximations of the Gaussian PDF and CDF[1].

Equation 12.5 and Equation 12.12 are thus equivalent and give the same answer. However, the boundary line integral is more efficient to compute than a surface integral. Also, because the CDF is used, the variance in the result is minimally by the glossiness of the surface.

Given a beam segment, Equations 5.3 to 5.8 still describe how to calculate the $\vec{\phi}_i$ vectors required for the domain of Equation 12.12. Section 12.5

[1]Recall that the Gaussian PDF stems from the spherical Gaussian glossy BRDF derived in Chapter 12.

showed that these equations are potentially also applicable to diffuse scattering surfaces if the surface radiance distribution is known.

It is important to realise that the forward rendering time of LBT is strongly dependent on the glossiness of the materials. This is due to the overlap of the glossy beams. Typically for $\sigma = 0.05$, 60% of the execution time is spent during forward rendering traversing the beam BVH in search of glossy beams that potentially contribute to a point. The efficiency of the initial beam tracing phase does, however, mean that one could spend more time building more efficient light beam hierarchies (or other light field representations) to improve the forward render time.

Potentially, further significant speedup of LBT could for example still be attained by merging neighbouring beam path segments that are similar to reduce the number of light beams. The boundary line solution also favours merged beams, due to the improved circumference to surface ratios of merged beams. Future work could further focus on replacing the numerical line integral with a table lookup or a piecewise analytical solution.

Comparison has not yet been done with other state-of-the-art rendering algorithms for caustics, such as using photon differentials and photon relaxation. These algorithms are extensions of PM and hence still—like classical PM—are required to trace a far greater number of photons than the number of beams required by LBT. A detailed comparison including an appropriate image similarity metric could, however, more informatively guide the development of beam tracing.

Part V

About the Code

This part of the thesis discusses some important parts of the software code that was developed during the PhD study.

Chapter 16

StitchEngine Overview

The StitchEngine is a by-product of my PhD research. The main objective for creating it was to provide a running code base with examples to accompany my research outputs. As mentioned, it is available at: `http://code.google.com/p/stitch-engine/source` tag sprinkles[1]. According to data generated using David A. Wheeler's `SLOCCount` the software contains approximately 16k lines of code.

The project optionally depends on:

- OpenSceneGraph, for result display and preview rendering as well as loading surface textures.
- OpenEXR, for saving of radiance map (a.k.a. HDR) images.
- Boost, for threading and random number generation when C++11 is not available.

The CMake build system is used and most of the development was done in Xcode and QtCreator under OSX and in QtCreator under Debian Linux with GCC. To create an Xcode project for example, from the terminal in a new `build` folder simply execute `cmake -G Xcode PATH_TO_CMAKELISTS_TXT` and then execute `open StitchEngine.xcodeproj/`.

The Doxygen documentation is no longer included in the svn repository, but it is one of the compile targets of the cmake project. To build the

[1]The motivation for the tag name is given in `CoffeeShake.txt` in the code repository.

documentation simply run `make Doxygen` or build the Doxygen target from your IDE.

To guide the reader through the source, an overview of the code and the class hierarchy is given next in Section 16.1. Section 16.2 describes the optimisation strategies. The BVH implementations are then discussed briefly in Section 16.3. Finally, chapter 17 gives an overview of the renderer implementations. The Doxygen documentation also captures the list of @todo comments left throughout the code. Figure 16.1 shows the list of implemented classes.

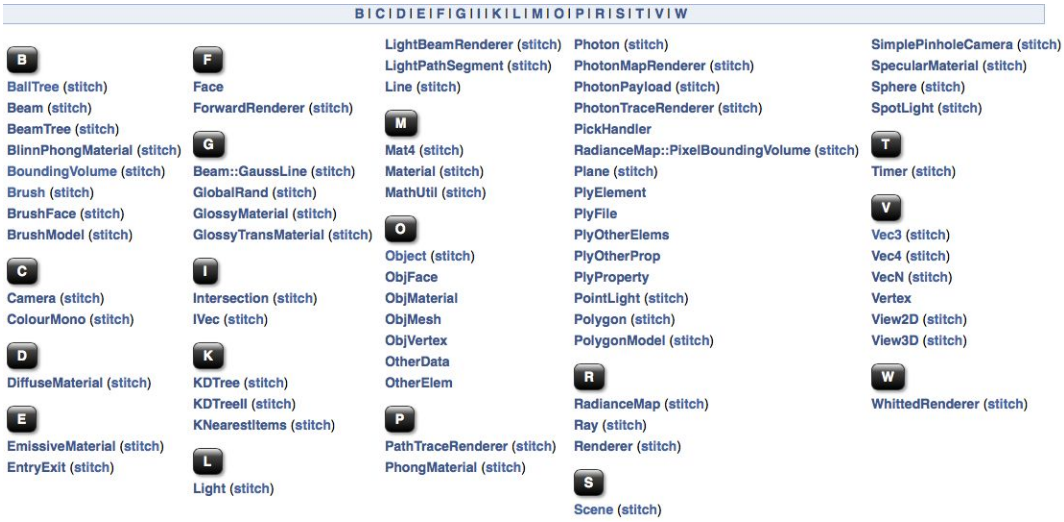


Figure 16.1: List of software classes.

16.1 Call Graphs and Class Hierarchies

Partial class hierarchies and call graphs are shown to communicate the design of the software. This is intended as a starting point for understanding and navigating the Doxygen documentation. The diagrams that are shown are taken from the generated documentation. The `CMakeLists` file contains a Doxygen target which may be selected if Doxygen is installed.

Once the scenario is loaded, an image may be rendered by calling the Renderer's `render` method. The LBT example application starts a new thread,

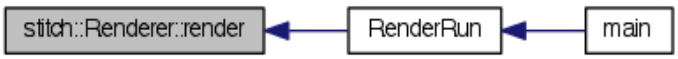


Figure 16.2: Calling of the renderer's `render` method.

`RenderRun`, to do so. Figure 16.2 shows the call graph.

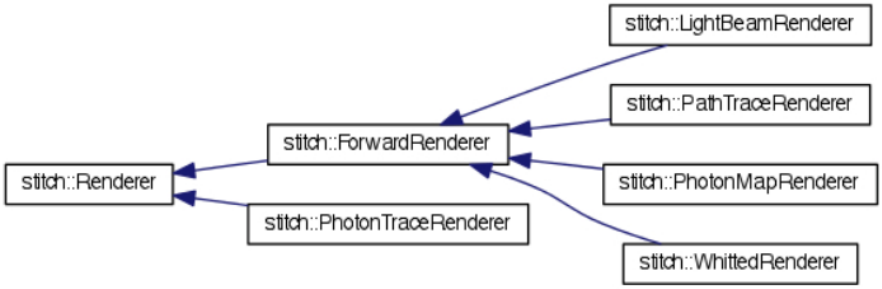


Figure 16.3: The class hierarchy of renderers.

Figure 16.3 shows the class hierarchy of renderers implemented. Most renderers fall in the forward rendering category because they include at least one forward render pass.

Figure 16.4 shows the `render` and `renderTask` call graphs. Calling a renderer's `render` method spawns a thread for each processor core in the system which each then runs a `renderTask`. Each render task renders part of the radiance map frame buffer making use of the camera's `getPrimaryRay` method and the specific renderer's (`LightBeamRenderer` in this case) `gather` method.

The preview display is updated from the main thread every second or so. Note that the frame buffer locations are rendered in a random shuffled order which allows for a quicker preview of the result. When the Voronoi display preview is used the un-rendered pixels are set to the colour of their nearest neighbour as shown in Figure 16.5.

An efficient concurrent radiance map frame buffer is required to support the random shuffled frame buffer. The buffer is accessed from multiple parallel render tasks as well as the preview display thread. The concurrent radiance map interface methods are `setMapValue(...)`, `addToMapValue(...)`,

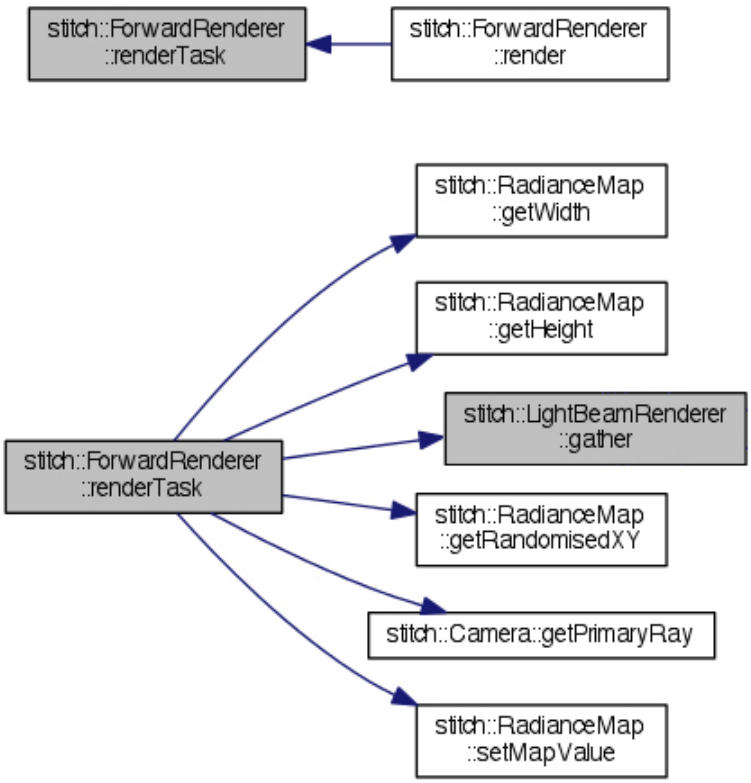


Figure 16.4: The `render` and `renderTask` call graphs.

`updateDisplayBuffer(...)` and `updateVoronoiDisplayBuffer(...)` in `RadianceMap.h`. The renderers' code may be inspected to see how these are used.

Figure 16.6 shows the class hierarchy of scene objects. All objects inherit from the base bounding volume class which gives it a centre and bounding volume radius. Notice that the `BallTree` (StitchEngine's scene BVH), the `Photon` and the radiance map's pixel all also inherit from the bounding volume base class. The objects implemented for the thesis were a simple point light, a Brush (k-DOP), a BrushModel (ball tree of k-DOPs), a Polygon and PolygonModel (effectively a ball tree of polygons) and, of course, a sphere. The k-DOP BVH is discussed further in Section 16.3.

Figure 16.7 shows the class hierarchy of material BRDFs. The Glossy-

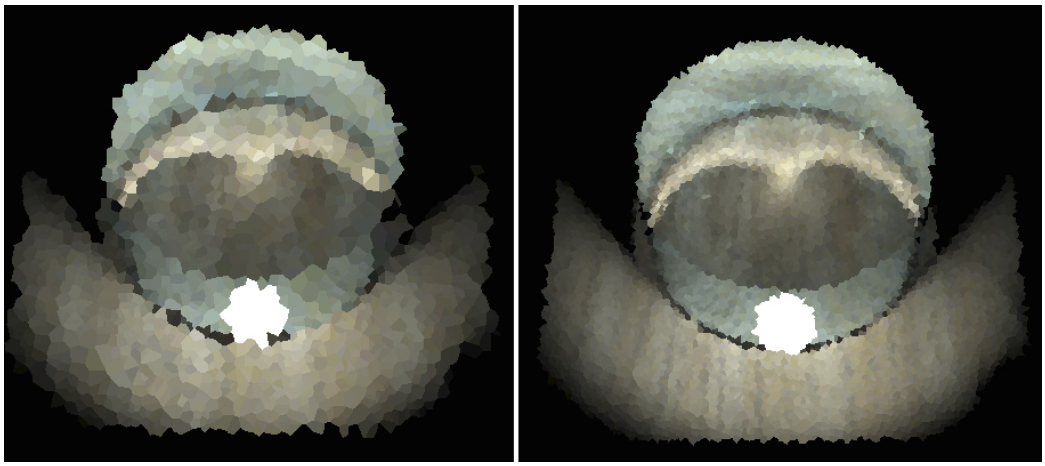


Figure 16.5: Voronoi display preview at 1% and 5% render progress.

Material implements the BRDF derived in this thesis. All the other material BRDFs are also the implementations adapted for physical plausibility as described in Part III of the thesis.

The BRDF interface methods that are of greatest importance are the `scatterPhoton_direct(...)`, `BRDF(...)` and `scatterPhoton_reject_samp(...)` methods. The first two methods implement the specific BRDFs' characters while the third is actually a base class method that uses rejection sampling and the virtual `BRDF(...)` method to generate random scattered photons using the appropriate PDF.

The random vector distributions required for the BRDFs and other operations are implemented within the `Vec3` class within methods such as `randBall()`, `randNorm()` and `randCosineLobe()` that return an appropriately distributed random vector with each call. Each random vector generator also has an associated PDF such as `randCosineLobe_pdf(const Vec3 &v)` to sample the PDF for a particular vector `v`. The implementation of these distributions may be found in `Math/Vec3.cpp`. The `randGaussianLobe()`, for example, makes use of the Box-Muller transform to generate a Gaussian distributed random vector from two uniform random numbers.

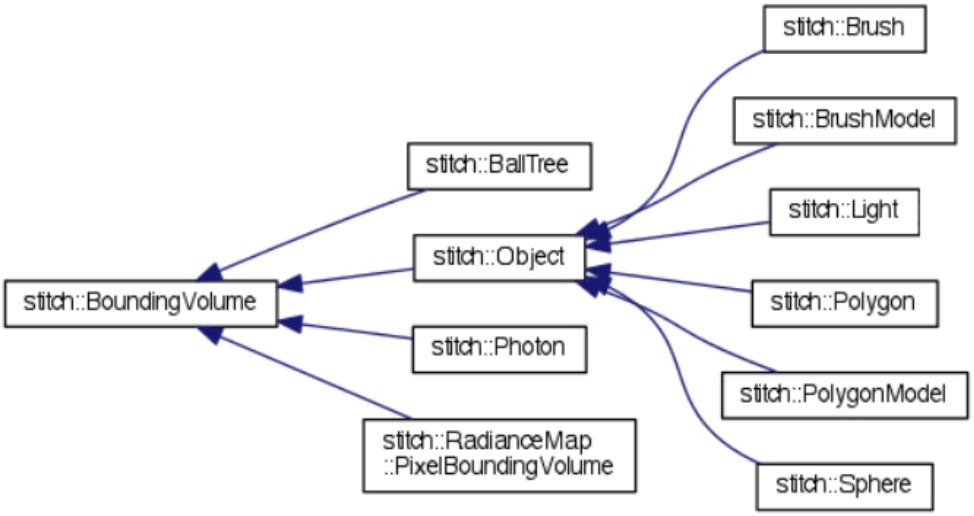


Figure 16.6: The class hierarchy of scene objects

16.2 Optimisation Strategies

I generally attempted to use C++11/STL best practices, efficient algorithms and simple code. I kept the CPU cache in mind to some extent, but I let the compiler worry about the machine code and the low-level optimisation. I found Scott Meyers' books [71][72] to be very valuable as well as the numerous talks on Microsoft's Channel 9 on C++ and C++11.

The C++ best practices include things such as:

- Use pre-increment instead of post-increment, especially for non-primitive types.
- For C++98, appropriately using sqrtf as opposed to sqrt, for example, and generally being mindful of static type conversion/casting proved valuable.
- Placing the `explicit` keyword in front of often used constructors and conversion operators to have the compiler flag any hidden potentially expensive conversion constructors being called.
- I relied on return value optimisation and move semantics to have simple, but fast code.

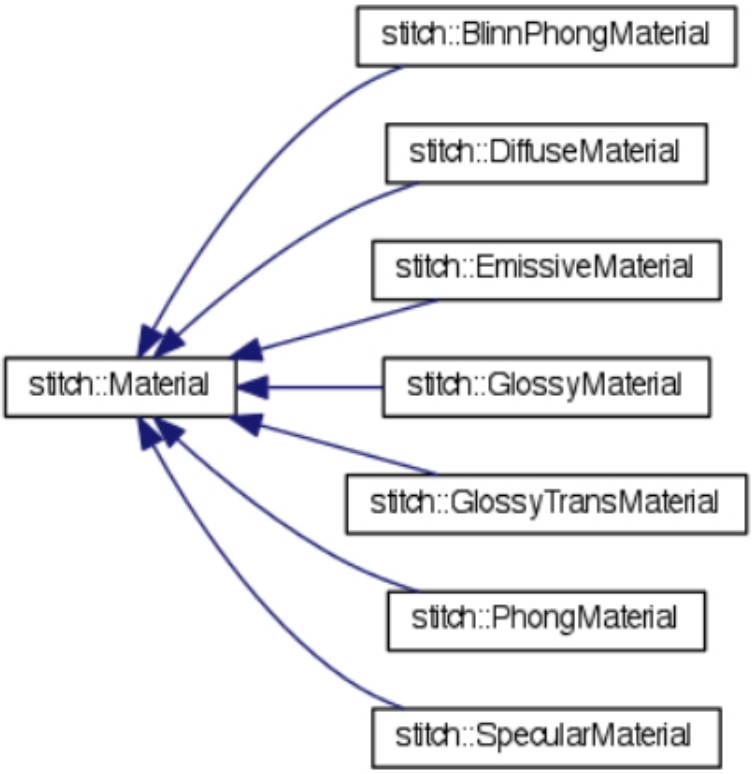


Figure 16.7: The class hierarchy of material BRDFs

- Use of `std::mt19937_64` random generator along with the appropriate `std` distribution.

I did some experimentation with hand generated single instruction multiple data (SIMD) intrinsics in the `Vec3` class, but generally relied on the compiler to optimise such vector operations for 4 wide SIMD.

Currently, the code is not designed to exploit wider than 4 wide SIMD instructions. However, Intel's Embree [73] API is an open source ray tracing API optimised and designed for SIMD packet tracing on modern Intel CPUs including the Xeon Phi. I have found it to be a good example of optimised code. Of course, in future the Embree API could also used be instead of the current StitchEngine ray tracing code.

16.3 The Bounding Volume Hierarchies

Two bounding volume hierarchies are implemented. A ball-tree for the scene BVH and a k-DOP beam-tree for the light beam segment BVH.

The ball-tree is implemented in `BallTree`. The current implementation is still a pointer heavy implementation and therefore cache unfriendly. An array based compact and pointer-less version is left as future work.

Each ball-tree node is a bounding volume of its list of items[2] and its list of child trees. When building the tree one usually starts with one node that contains all the items and then recursively splits the items into child nodes. The list of items is split by a splitting plane through the centroid of the items. The splitting plane orientation to use at each level in the tree is passed as a parameter to the `build` method. Usually one would split the scene along one of its three dimensions viz. along x then along y then along z then along x again and so on. Splitting is continued until the number of items in a node falls below a threshold. An improved split metric such as the surface area heuristic (SAH) discussed by Pharr and Humphreys [10] has not yet been implemented.

The k-DOP beam-tree is implemented in `BeamTree`. It is used exclusively for the light beam segment BVH of the LBT renderer.

Each beam-tree node is a bounding volume for its list of beam segments and its list of child trees. The beam-tree node has a k-DOP bounding volume member. The k-DOP is implemented by the `Brush`[3] class. The beam segment is implemented by the (BeamSegment) class which also has a k-DOP bounding volume member. Each beam has an end-cap polygon formed by the intersection of the beam corners with the receiving surface. The tree is built in a similar fashion to the ball-tree, but the splitting is done based on the centroids of the end-caps.

[2] An item is any C++ object of base class BoundingVolume.

[3] A brush is the intersection of a set of half-spaces and a synonym for convex 3D polytope or k-DOP.

Chapter 17

The Renderer Implementations

Three rendering algorithms were implemented for use in the thesis. The light trace renderer [3] is a photon simulation with only a light pass. The photon map renderer and the light beam renderer have a light phase followed by a forward camera phase and are therefore further specialisations of the ForwardRenderer class. The renderer class hierarchy is shown in Figure 16.3.

Similar to the other chapters in this part of the thesis, this chapter should guide the reader through the source code. It is not meant to also give detailed explanations of the light trace and photon map renderers. The books by Dutré et al. [15] and Jensen [74] offer detailed explanations of these rendering algorithms.

Every time a frame is rendered a new instance of the specified renderer is created in the `RenderRun()` function in `main.cpp` which was mentioned earlier. The renderer is instantiated with a pointer to the scene, while the `render` method takes as parameters the radiance map render target, the camera and the frame duration.

17.1 The Light Trace Renderer

The light trace renderer [3] is a photon simulation with only a light pass. It is implemented in `PhotonTraceRenderer.h` and `PhotonTraceRenderer.cpp`. Photon tracing is used synonymously with light tracing.

The `render` method does a number of light trace iterations to limit the size of the `inFlightPhotonVector_` required to hold the photons being traced. The `tracePhotons` method calls `radiate` on the scene light source and then traces the photons through the scene. At each photon-scene intersection a photon is also directly traced to the focal plane. If this *direct* photon is not occluded then it is stored in the `photonVector_` to be deposited in the radiance map at the end of the light trace iteration.

Each light trace iteration ends by adding the radiance from the photons in `photonVector_` to the radiance map. The `radianceMap.addToMapValue` method does not yet support the Voronoi frame buffer preview.

17.2 The Light Beam Renderer

The light beam renderer has a light phase followed by a forward camera phase. It is therefore a further specialisations of the `ForwardRenderer` class. It is implemented in `LightBeamRenderer.h` and `LightBeamRenderer.cpp`.

The `render` method is inherited from the `ForwardRenderer` parent. The `ForwardRenderer` children need only implement the `preForwardRender` and `gather` methods.

The `preForwardRender` method simply calls the light beam renderer's `traceBeams` method which implemented the light phase and light image mesh processing.

The `gather` method implements the gather operation of the forward rendering phase. This method calls the `BVH.getContributingBeamSegments(...)` and `calcFluxProb(...)` methods of Algorithm 2 as described in Chapter 12. The basic rendering algorithm is a Whitted forward renderer with a radiance contributions due to the light beams.

17.3 The Photon Map Renderer

The photon map renderer has a light phase followed by a forward camera phase. It is therefore a further specialisation of the ForwardRenderer class.

It is implemented in `PhotonMapRenderer.h` and `PhotonMapRenderer.cpp`.

As is the case for `LightBeamRenderer`, the `render` method is inherited from the `ForwardRenderer` parent. The `ForwardRenderer` children need only implement the `preForwardRender` and `gather` methods.

The `preForwardRender` method simply calls the light beam renderer's `tracePhotons` method which implemented the light phase, photon tracing and building of the photon map kd-tree.

The `gather` method implements the gather operation of the forward rendering phase. A direct photon radiance estimate is implemented in this method. Final gathering is not used because the renderer is used for rendering caustics. The basic rendering algorithm is also a Whitted forward renderer with a radiance contribution from the photon map.

Part VI

Conclusion and Future Work

This part concludes the thesis. It critically reflects on the success of the undertaking, highlights limitations and points ahead to future work.

Chapter 18

Conclusion

The expectation is that the proposed irradiance solution and the rendering algorithm (or a derivative thereof) would in future be useful for more general applications. However, one of the trends in the state of the art of rendering is to use multiple estimators (i.e. combine multiple rendering algorithms) and to know when to use which. Glossy LBT could therefore be applied as an alternative to render LG+D transport paths while the transport paths not suitable to LBT are rendered by other rendering algorithms. Please see the Related Works chapter, Chapter 3, for state of the art rendering algorithms for caustics.

The glossy BRDF that is required to support glossy LBT was derived and verified to be physically plausible. The glossy BRDF equation is Equation 8.3:

$$f_r(y, \Psi', \vec{\phi}) \quad = \quad \frac{\rho_s e^{-\frac{\phi^2}{2\sigma^2}}}{2\pi\sigma^2 |\cos(N_y, \Psi')|}$$

Wielding Gauss' divergence theorem, the irradiance due to a glossy beam segment may be efficiently calculated by using Equation 12.12:

$$E\left(x\right) \quad \approx \quad \rho_s \frac{\Phi_s}{A_{x\perp}} |\cos\left(N_x, \overline{\Psi}\right)| \oint_{AB,BC,CA} \vec{F} \cdot \vec{n} \; ds_\phi.$$

Given a beam segment, Equations 5.3 to 5.8 describe how to calculate the $\vec{\phi}_i$

vectors required for the domain of integration. AB, BC, CA are simply the boundary lines $\vec{\phi_0} \rightarrow \vec{\phi_1}$, $\vec{\phi_1} \rightarrow \vec{\phi_2}$ and $\vec{\phi_2} \rightarrow \vec{\phi_0}$.

I apply Gauss' divergence theorem to the spherical Gaussian light beam which is much different from Xu et al' approach or anything proposed earlier [51][75] on integration of the spherical Gaussians used in rendering of caustics.

As mentioned, the study into LBT and the proposed light beam radiance estimate should in no way conclude with this thesis. For example, the assumptions that lead to Equation 12.12 are regarding the approximately locally constant $\cos(N_y, \Psi')$ and $\cos(N_x, \Psi)$, the use of the glossy BRDF which is only symmetric for relatively glossy materials as well as the assumption that Equation 12.4 holds for glossy beams. Beyond the assumptions there are also other remaining challenges in applying the irradiance equations in practice:

- The LBT irradiance equations are currently only useful for symmetric scatter lobes. However, many BRDFs such as specular microfacet BRDFs, result in asymmetrical scatter lobes and the asymmetry usually increases with incident angle.

- It does remain a challenge to render light beams more efficiently than rendering an *already built* kd-tree photon map. The execution performance is linearly dependent on the number of beams returned from the BVH that potentially contribute irradiance to a point. Especially in areas where many beams overlap does this become a problem.

- The multi-bounce example uses the convolution of spherical Gaussian scatter lobes to calculate the effective scatter lobe along a beam's path. For this reason the multi-bounce LBT described should be limited to beam geometries that are relatively wide compared to the scatter lobes. For narrower beams tracing light rays would be more suitable.

- For all its benefits, the dynamic refinement of the light image approach unfortunately results in many small beams that contribute to the last two challenges mentioned above.

However, the better forward render times and the relatively efficient light phase make LBT a good alternative to PM. The LBT algorithm also has a very low memory footprint, due to the relatively small number of light rays and light beams that are required. As a result, special measures to content with memory consumption, such as, for example, a progressive rendering solution, was therefore not needed.

Significant speedup of LBT could potentially be attained by re-merging neighbouring similar beam path segments to reduce the number of overlapping beams. The boundary line solution also favours merged beams, due to the improved circumference to surface ratios. Furthermore, the boundary line solution applies to beams with three, four or more corners. The efficiency of the beam tracing phase also means that one could in future spend more time building better light beam hierarchies (or other light field representations) to improve the forward render time. The very first research papers [53][54] on glossy LBT indeed showed that if the number of overlapping beams is kept small then the proposed technique is suitable for interactive and real-time image synthesis.

Although transparent materials have not been included since the research paper [54] published in 2011, refractive surfaces as well as area lights are straightforward extensions of the rendering algorithm. The demonstration of such wider application of the rendering algorithm is left as future work. Further, as discussed, potential dualities exist between glossy LBT, Arvo's irradiance tensor work [21], Xu et al.'s all-frequency render cut algorithm [52] and even Lambert's diffuse irradiance / form-factor equation (Eq 12.13). Future work should certainly explore this in more depth.

Chapter 19

Future Work

I plan to employ Equation 12.12 in combination with other scene radiance distributions or light field representations. Better quantification of the various approximations mentioned during the derivation will however be important and should be investigated further. A related future topic for research could be to evaluate other approximations of the Gaussian PDF and CDF[1].

It should also be possible to extend the rendering algorithm to asymmetric scatter lobes—when using other BRDFs such as the Blinn-Phong and Torrance-Sparrow BRDFs—and different lobe widths per beam corner. It should in fact be possible to do the entire derivation with the radiance distribution instead of the scatter distribution. The challenge would be in still employing Gauss' divergence theorem.

It is worth investigating alternatives to the beam hierarchy for representing the light envelopes. It could also be possible to apply Equation 12.12 to light field representations such as photon maps, Havran et al.'s [76] ray maps and Kaplanyan et al.'s [77] Light Propagation Volumes.

Future work could also centre around:

- a piecewise analytical solution or a table lookup for the line integral;
- including improved shadow ray sampling;

[1]Recall that the Gaussian PDF stems from the spherical Gaussian glossy BRDF derived in Chapter 12.

- glossy receiving surfaces;
- fixing of light image mesh cracks;
- rendering of refractive surfaces;
- including area lights; and
- rendering of participating media.

What is more, as noted, it would be worthwhile to investigate the post light phase merging of neighbouring light image mesh elements to reduce the number of overlapping beams. The boundary line solution also favours merged beams due to the improved circumference to surface ratios and the solution applies to beams with three, four or more corners. There might also be graph-cut-like optimisations to LBT to explore, such as using beam-tree non-leaf nodes for approximate illumination. Further investigation of potential dualities between Equation 12.12 derived in this thesis, Lambert's Equation (Eq 12.13), Arvo's irradiance tensors [21] and Xu et al.'s [52] work could also be valuable.

Comparison has not yet been done with other state-of-the-art rendering algorithms for caustics, such as using photon differentials and photon relaxation. These algorithms are extensions of PM and hence still—like classical PM—are required to trace a far greater number of photons than the number of beams required by LBT. A detailed comparison including an appropriate image similarity metric could, however, more informatively guide the development of beam tracing.

Future work could additionally shift focus towards using alternative hardware and/or software platforms such as, for example, the Open Compute Language (OpenCL), NVidia's CUDA or the Intel SPMD Program Compiler (ICMP) to create implementations for execution on GPUs and other many core processor architectures.

Chapter 20

Poster

I have prepared a poster on my thesis. I include it here for the readers that would like a one page overview of the research.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Physically Based Rendering and Light Beam Tracing

Bernardt Duvenhage [PhD Supervisors: Prof DG Kourie (SUN) and Prof K Bouatouch (Rennes 1)]
bduvenhage@csir.co.za
CSIR, Defence Peace Safety and Security (DPSS), South Africa

CSIR our future through science

Overview

The physics of light transport have been brought together in Kajiya's [4] rendering equation. The algorithms that have been designed to solve the rendering equation are known as *rendering algorithms*.
Light beam tracing (LBT) is a two pass bi-directional rendering algorithm which is able to exploit the coherency of the transport paths within an envelope of light. LBT rendering algorithms found in the literature unfortunately only support mirror-like specular surface interactions.

Brief Scope

This thesis describes an extension of LBT to also include glossy surface interactions. The scope of the work encompasses both the derivation of the new physically based rendering algorithm and a multi-core CPU implementation.
The software that was written for the thesis is available at: `http://code.google.com/p/stitch-engine/source` tag sprinkles[a].

[a] The motivation for the tag names is given in `CoffeeShake.txt` in the code repository.

Contributions

This thesis makes several contributions:

- A new physically plausible spherical Gaussian BRDF is formulated.
- Single bounce LBT is extended to include glossy surface interactions, thus broadening the application domain of LBT.
- Gauss' divergence theorem is used to replace the surface integral of the rendering equation with a boundary line integral. This enhances[a] both the *quality and performance* of single bounce glossy LBT. As far as I know this is the first application of Gauss' divergence theorem to solve the lighting integral for rendering caustics.
- A light image is used to extend glossy LBT to *multi-bounce* transport paths.

My paper [3] discusses the thesis and contributions in more detail.

[a] One might tongue-in-cheek say "Stokes".

1. Conjecture

Using the Dirac Delta to reformulate the equation for the irradiance due to a beam scattered from a specular surface:

$$E(x) = \frac{\rho_s \Phi_s}{A_{x\perp}} |\cos(N_x, \overline{\Psi})| \iint_{\Omega} \delta(\vec{\phi}) d\omega_{\phi}. \quad (1)$$

lead to the *conjecture* that the irradiance due to a light beam scattered from a glossy surface may be calculated by replacing the Dirac Delta with a PDF ρ that has larger support:

$$E(x) = \frac{\rho_s \Phi_s}{A_{\perp}} |\cos(N_x, \overline{\Psi})| \iint_{\Omega_{ABC}} \rho(\vec{\phi}) d\omega_{\phi}. \quad (2)$$

2. The Glossy BRDF

The Dirac Delta and ρ PDFs embody the lobe shape of the scattered light and a spherical Gaussian definition of ρ was used:

$$\rho(\phi) = \frac{1}{2\pi\sigma^2} e^{-\frac{\phi^2}{2\sigma^2}} \quad (3)$$

ϕ is the angle between the scattered light and the mirror-like scatter direction.
A BRDF that exhibits this lobe shape is required to support the conjecture and it just so happens that such a BRDF is possible *and* physically plausible:

$$f_r(y, \Psi', \phi) = \left[\frac{\rho_s}{2\pi\sigma^2} e^{-\frac{\phi^2}{2\sigma^2}}\right] \frac{1}{|\cos(N_y, \Psi')|} \quad (4)$$

3. Gauss' Divergence Theorem

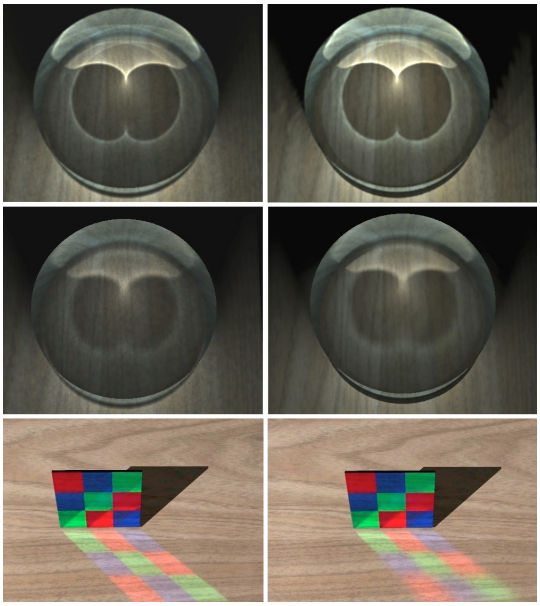
It turns out that one *can* derive the conjectured equation from the rendering equation and the new glossy BRDF:

$$E(x) = \frac{\rho_s \Phi_s}{A_{\perp}} |\cos(N_x, \overline{\Psi})| \iint_{\Omega_{ABC}} \rho(\vec{\phi}) d\omega_{\phi}. \quad (5)$$

The presence of the surface integral is still a computational inconvenience. However, applying Gauss' divergence theorem, one may replace the surface integral with a much more efficient line integral:

$$E(x) \approx \frac{\rho_s \Phi_s}{A_{x\perp}} |\cos(N_x, \overline{\Psi})| \oint_{S_{ABC}} \vec{F} \cdot \vec{n}\, ds_{\phi}. \quad (6)$$

Results



Analysis

It does remain a challenge to render light beams more efficiently than rendering, for example, an *already built* kd-tree photon map. However, the comparative or better forward render times and the relatively efficient light phase makes LBT a good alternative to other caustics rendering algorithms.
Compared to photon mapping, the images shown on the left rendered in about half the time (for specular beams) to in the same time (for glossy beams). The more specular and glossy surfaces in the scene the more benefit is gained by LBT's efficient light phase and low memory footprint.

Conclusion & Future Work

Wielding Gauss' divergence theorem the irradiance due to a glossy beam segment may be efficiently calculated by using Equation 6. It is likely possible to also apply this equation to *other existing light field representations* such as Kaplanyan et al.'s [5] light propagation volumes. Further, post light phase merging of neighbouring light beams will reduce the number of overlapping beams and the boundary line solution favours merged beams due to the decreasing boundary to surface ratios. The early research papers [2] indeed showed that the proposed technique is suitable for interactive and real-time application.
Further investigation of potential dualities between Equation 6 derived in this thesis, Lambert's equation for irradiance due to a diffuse polygon luminaire, Arvo's irradiance tensors [1] and Xu et al.'s [6] work could also be valuable.

References

References

[1] James Arvo. Applications of irradiance tensors to the simulation of non-lambertian phenomena. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 1995. ACM.

[2] B. Duvenhage, K. Bouatouch, and D. G. Kourie. Extending backward polygon beam tracing to glossy scattering surfaces. *Computer Graphics Forum*, 30(6):1825–1836, 2011.

[3] B. Duvenhage, K. Bouatouch, and D. G. Kourie. Light beam tracing for multi-bounce specular and glossy transport paths. In *Proceedings of the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology*, SAICSIT '14, pages 199:199–199:208, New York, NY, USA, 2014. ACM.

[4] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[5] Anton Kaplanyan and Carsten Dachsbacher. 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, pages 99–107, New York, NY, USA, 2010. ACM.

[6] Kun Xu, Yan-Pei Cao, Li-Qian Ma, Zhao Dong, Rui Wang, and Shi-Min Hu. A practical algorithm for rendering interreflections with all-frequency brdfs. *ACM Trans. Graph.*, 33(1):10:1–10:16, February 2014.

Glossary

A list of common terms and acronyms is given below. The page number indicates where it is used and more information is available.

all-frequency All-frequency BRDFs, for example, include all BRDFs from specular to glossy and diffuse. 40

backward ray tracing A Monte Carlo photon simulation rendering algorithm by Arvo [4] that stores the photons deposited onto diffuse surfaces in an illumination map texture. 33

backward rendering The term *backward rendering* and *backward tracing* is used interchangeably with light tracing to describe ray or beam tracing *from the light* in the same direction as the flow of radiance solving the rendering equation using transport paths from the light source. 16

bi-directional rendering Bi-directional rendering refers to partially solving the rendering equation using transport paths from both the light source(s) and the camera then bringing the solutions together somewhere in between. 27

caustic The envelopes of light that are scattered from shiny curved surfaces and focussed into lines or spots of concentrated light. 15

diffuse surface A matte surface abiding by Lambert's cosine law viz. appears equally bright from all visible orientations. 24

flux The amount of energy per unit time that flows through a real or virtual boundary surface. 20

forward rendering Forward tracing and forward rendering is used to describe ray or beam tracing from the camera in the same direction as the flow of importance solving the rendering equation using transport paths from the camera. 16

geometric optics In terms of this assumption, light is considered to travel along straight lines as if consisting of particulate photons. 19

glossy beam A glossy light beam describes a light beam scattered from a glossy surface. 48

glossy surface A glossy surface is a roughened specular surface that results in blurry or fuzzy reflections. 25

importance sampling In statistics, importance sampling is a general technique for estimating properties of a particular distribution, while only having samples generated from a different distribution rather than the distribution of interest. 82, 129

irradiance The flux per unit area incident on a surface at a point. 20

kernel density estimate An estimate of a function value from discrete samples by convolution with a kernel function such as a Gaussian. 37

lambertian An ideal scattering or emitting surface that abides by Lambert's cosine law viz. appears equally bright from all visible orientations. 24

light beam An envelope of somewhat coherent light. The envelope is typically bounded by a number of corner light rays. 48

light beam tracing A bi-directional rendering algorithm that, during the light phase, traces envelopes or beams of light from the light source. A beam is defined by its corner light rays. 48

light image A raster based or mesh based representation of an image like projection of the scene, but from the light's point of view. 38

light tracing A Monte Carlo photon simulation rendering algorithm discussed by Dutré [15]. Typically a steady state or direct contribution to the camera is calculated. 28

Monte Carlo Monte Carlo *methods* or rendering algorithms rely on *random sampling* to numerically solve a problem such as the rendering equation. 27

multi-bounce A multi-bounce caustic is due to transport paths including one or more specular or glossy interactions viz. $L(S|G)^+D$. 28

photon mapping A bi-directional rendering algorithm that, during the light phase, traces individual photons of light from the light source. A photon is a packet of wideband or spectral energy. 29, 37

progressive rendering The image resolution and/or quality is progressively refined. A progressive renderer can render an image while staying within some resource budget such as limited memory or a maximum wall-clock frame rendering time. 38

radiance The flux per unit projected area per solid angle leaving or passing through a point on a surface in a certain direction. 22

radiance estimate The part of the rendering algorithm to estimate the radiance is often referred to simply as the radiance estimate. 37

random vector distribution A probability density function that describes the probability of occurrence of 3D direction vectors based on the angle of separation relative to some reference vector. 82

rejection sampling Sampling of a random variable by uniformly sampling from the region under the variables PDF. 82, 87, 93, 100, 102, 104, 106, 159, *see* PDF

rendering algorithm The algorithms that have been designed to solve the rendering equation are known as *rendering algorithms*. 16

rendering equation The physics of light transport have been brought together in Kajiya's [2] rendering equation. 16

second-bounce A second-bounce caustic is due to transport paths that have encountered *two* specular or glossy surface viz. $L(S|G)^2D$. 28

single-bounce A single-bounce caustic is due to transport paths that start at the light and encounter *one* specular or glossy surface interaction before encountering a diffuse receiving surface. Such a path may be written as $L(S|G)^1D$. 28

solid angle A two dimensional angle expressed in steradian (sr). 21

specular beam A specular light beam describes a light beam scattered by a mirror-like specular surface. 48

specular surface Mirror-like smooth surface. 24

spherical Gaussian A type of spherical radial basis function (SRBF) using a Gaussian function. 29

whitted ray tracing A forward ray trace rendering algorithm discussed by Whitted [17]. 33

Acronyms

BBT Backward beam tracing. 17

BVH Bounding volume hierarchy. 37

CBVH Cone bounding volume hierarchy. 51

CPU Central processing unit. 17

GPU Graphical processing unit. 30, 38

ISG Integral spherical Gaussian. 40

kNN k-nearest neighbours. 65

LBT Light beam tracing. 15

LT Light tracing. 28

MC Monte Carlo. 27

PDF Probability density function. 25, 56

PM Photon mapping. 29, 37

SG Spherical Gaussian. 29

SRBF Spherical Radial Basis Function. 29, 182

Bibliography

[1] Paul Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 145–154, New York, NY, USA, 1990. ACM.

[2] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[3] P. Dutré, K. Bala, and P. Bekaert. *Advanced global illumination.* Ak Peters Series. AK Peters, 2006.

[4] James Arvo. Backward ray tracing. In *Developments in Ray Tracing, SIGGRAPH Course Notes, Vol. 12.*, New York, NY, USA, 1986. ACM.

[5] Mark Watt. Light-water interaction using backward beam tracing. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 377–385, New York, NY, USA, 1990. ACM.

[6] Donald P. Greenberg. A framework for realistic image synthesis. *Commun. ACM*, 42(8):44–53, August 1999.

[7] F. E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Appl. Opt.*, 4:767–773, 1965.

[8] Johann Heinrich Lambert. *Photometria.* 1760.

[9] Eric Veach. *Robust monte carlo methods for light transport simulation.* PhD thesis, Stanford University, Stanford, CA, USA, 1998. AAI9837162.

[10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[11] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[12] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, 1977.

[13] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, 1982.

[14] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental validation of analytical brdf models. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, pages 90–, New York, NY, USA, 2004. ACM.

[15] Philip Dutré, Eric P Lafortune, and Yves Willems. Monte carlo light tracing with direct computation of pixel intensities. In *3rd International Conference on Computational Graphics and Visualisation Techniques*, pages 128–137, 1993.

[16] Eric Lafortune. *Mathematical models and Monte Carlo algorithms for physically based rendering*. PhD thesis, Citeseer, 1996.

[17] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23:343–349, June 1980.

[18] Paul S. Heckbert and Pat Hanrahan. Beam tracing polygonal objects. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 119–127, New York, NY, USA, 1984. ACM.

[19] Tomoyuki Nishita, Yasuhiro Miyawaki, and Eihachiro Nakamae. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. *SIGGRAPH Comput. Graph.*, 21(4):303–310, 1987.

[20] Mikio Shinya, Takafumi Saito, and Tokiichiro Takahashi. Rendering techniques for transparent objects. In *Graphics Interface*, pages 173–182, 1989.

[21] James Arvo. Applications of irradiance tensors to the simulation of non-lambertian phenomena. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 1995. ACM.

[22] Don Mitchell and Pat Hanrahan. Illumination from curved reflectors. *SIGGRAPH Comput. Graph.*, 26(2):283–291, July 1992.

[23] Tomoyuki Nishita and Eihachiro Nakamae. Method of displaying optical effects within water using accumulation buffer. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 373–379, New York, NY, USA, 1994. ACM.

[24] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995.

[25] Henrik Wann Jensen. Rendering caustics on non-lambertian surfaces. In *Proceedings of the Conference on Graphics Interface '96*, GI '96, pages 116–121, Toronto, Ont., Canada, Canada, 1996. Canadian Information Processing Society.

[26] J. Chuang and S. Cheng. Computing caustic effects by backward beam tracing. *The Visual Computer*, 11(3):156–166, 1995.

[27] Normand Brière and Pierre Poulin. Adaptive representation of specular light. *Computer Graphics Forum*, 20(2):149–159, June 2001.

[28] K. Iwasaki, Y. Dobashi, and T. Nishita. An efficient method for rendering underwater optical effects using graphics hardware. *Computer Graphics Forum*, 21(4):701–711, 2002.

[29] K. Iwasaki, Y. Dobashi, and T. Nishita. A fast rendering method for refractive and reflective caustics due to water surfaces. *Computer Graphics Forum*, 23(3):601–609, 2003.

[30] Manfred Ernst, Tomas Akenine-Möller, and Henrik Wann Jensen. Interactive rendering of caustics using interpolated warped volumes. In *GI '05: Proceedings of Graphics Interface 2005*, pages 87–96, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[31] Lars Schjøth, Jeppe Revall Frisvad, Kenny Erleben, and Jon Sporring. Photon differentials. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 179–186, New York, NY, USA, 2007. ACM.

[32] Homan Igehy. Tracing ray differentials. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 179–186, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[33] Frank Suykens and Yves D. Willems. Path differentials and applications. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 257–268, London, UK, UK, 2001. Springer-Verlag.

[34] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. The beam radiance estimate for volumetric photon mapping. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, pages 3:1–3:112, New York, NY, USA, 2008. ACM.

[35] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–8, New York, NY, USA, 2008. ACM.

[36] Charly Collin, Mickaël Ribardière, Adrien Gruson, Rémi Cozot, Sumanta Pattanaik, and Kadi Bouatouch. Visibility-driven progressive volume photon tracing. *The Visual Computer*, 29(9):849–859, 2013.

[37] B. Spencer and M. W. Jones. Into the blue: Better caustics through photon relaxation. In *Eurographics 2009: Proceedings of the 30th Annual Conference of the European Association for Computer Graphics*, Munich, Germany, 2009.

[38] Ben Spencer and Mark W Jones. Photon parameterisation for robust relaxation constraints. In *Computer Graphics Forum*, volume 32, pages 83–92. Wiley Online Library, 2013.

[39] Ben Spencer and Mark W. Jones. Progressive photon relaxation. *ACM Trans. Graph.*, 32(1):7:1–7:11, February 2013.

[40] Lieu-Hen Chen, Tsung-Chih Tsai, and Yu-Sheng Chen. Grouped photon mapping. *Vis. Comput.*, 26(3):217–226, March 2010.

[41] Normand Brière and Pierre Poulin. Adaptive representation of specular light flux. In Sidney Fells and Pierre Poulin, editors, *Graphics Interface*, pages 127–136, Montreal, Canada, May 2000. Canadian Human-Computer Communications Society.

[42] Wei Hu, Zhao Dong, Ivo Ihrke, Thorsten Grosch, Guodong Yuan, and Hans-Peter Seidel. Interactive volume caustics in single-scattering media. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pages 109–117, New York, NY, USA, 2010. ACM.

[43] Gábor Liktor and Carsten Dachsbacher. Real-time volume caustics with adaptive beam tracing. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 47–54, New York, NY, USA, 2011. ACM.

[44] Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics (TOG)*, 30(1):5, 2011.

[45] Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. Progressive photon beams. In *ACM Transactions on Graphics (TOG)*, volume 30, page 181. ACM, 2011.

[46] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192:1–192:10, November 2012.

[47] Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph.*, 33(4):1–13, August 2014.

[48] Gregory J. Ward. Measuring and modeling anisotropic reflection. *SIGGRAPH Comput. Graph.*, 26(2):265–272, July 1992.

[49] Yu-Ting Tsai and Zen-Chung Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph.*, 25(3):967–976, July 2006.

[50] Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph.*, 28(5):133:1–133:10, December 2009.

[51] Kei Iwasaki, Wataru Furuya, Yoshinori Dobashi, and Tomoyuki Nishita. Real-time rendering of dynamic scenes under all-frequency lighting using integral spherical gaussian. *Comp. Graph. Forum*, 31(2pt4):727–734, May 2012.

[52] Kun Xu, Yan-Pei Cao, Li-Qian Ma, Zhao Dong, Rui Wang, and Shi-Min Hu. A practical algorithm for rendering interreflections with all-frequency brdfs. *ACM Trans. Graph.*, 33(1):10:1–10:16, February 2014.

[53] Bernardt Duvenhage, Kadi Bouatouch, and Derrick Kourie. Exploring the use of glossy light volumes for interactive global illumination. In *AFRIGRAPH '10: Proceedings of the 7th International Conference on*

Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, pages 139–148, New York, NY, USA, 2010. ACM.

[54] B. Duvenhage, K. Bouatouch, and D. G. Kourie. Extending backward polygon beam tracing to glossy scattering surfaces. *Computer Graphics Forum*, 30(6):1825–1836, 2011.

[55] Alfred S Posamentier and Ingmar Lehmann. *The secrets of triangles: a mathematical journey*. Prometheus Books, 2012.

[56] B. Duvenhage, K. Bouatouch, and D. G. Kourie. Numerical verification of bidirectional reflectance distribution functions for physical plausibility. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, SAICSIT '13, pages 200–208, New York, NY, USA, 2013. ACM.

[57] Eric P Lafortune and Yves D Willems. Using the modified phong reflectance model for physically based rendering. *Report CW*, 197:2–4, 1994.

[58] R. Sargent. An expository on verification and validation of simulation models. In *Proceedings of 1985 Winter Simulation Conference*, Orlando, Florida, USA, 1985.

[59] R. Sargent. Verification, validation, and accreditation of simulation models. In *Proceedings of 2000 Winter Simulation Conference*, Orlando, Florida, USA, 2000.

[60] B. Zeigler, T. Kim, and H. Praehofer. *Theory of Modelling and Simulation, second edition*. Academic Press, San Diego, California, USA, 2000.

[61] Ricardo Marques, Christian Bouville, Mickaël Ribardière, Luís Paulo Santos, and Kadi Bouatouch. Spherical fibonacci point sets for illumination integrals. In *Computer Graphics Forum*, volume 32, pages 134–143. Wiley Online Library, 2013.

[62] Robert R. Lewis. Making shaders more physically plausible. In *In Fourth Eurographics Workshop on Rendering*, pages 47–62, 1994.

[63] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[64] C. J. Willers. *Electro-Optical System Analysis and Design: A Radiometry Perspective*. SPIE PRESS, 2013.

[65] B. Duvenhage, K. Bouatouch, and D. G. Kourie. Light beam tracing for multi-bounce specular and glossy transport paths. In *Proceedings of the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology*, SAICSIT '14, pages 199:199–199:208, New York, NY, USA, 2014. ACM.

[66] David Jeffrey Griffiths. *Introduction to Electrodynamics, International Edition*. Pearson, third edition, 2008.

[67] DavidH. Raab and EdwardH. Green. A cosine approximation to the normal distribution. *Psychometrika*, 26(4):447–450, 1961.

[68] Tomoyuki Nishita and Eihachiro Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In *ACM SIGGRAPH Computer Graphics*, volume 19, pages 23–30. ACM, 1985.

[69] Michael M Stark and Richard F Riesenfeld. *Exact illumination in polygonal environments using vertex tracing*. Springer, 2000.

[70] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. In *IEEE Transactions on Visualization and Computer Graphics*, pages 21–36. IEEE Computer Society, 1998.

[71] Scott Meyers. *Effective STL: 50 specific ways to improve your use of the standard template library*. Pearson Education, 2001.

[72] Scott Meyers. *Effective Modern C++*. O'Reilly, 2014.

[73] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. Embree: A kernel framework for efficient cpu ray tracing. *ACM Trans. Graph.*, 33(4):143:1–143:8, July 2014.

[74] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. AK Peters, Ltd., 2001.

[75] Rui Wang, Minghao Pan, Weifeng Chen, Zhong Ren, Kun Zhou, Wei Hua, and Hujun Bao. Analytic double product integrals for all-frequency relighting. *Visualization and Computer Graphics, IEEE Transactions on*, 19(7):1133–1142, 2013.

[76] Vlastimil Havran, Jiří Bittner, and Hans-Peter Seidel. Ray maps for global illumination. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, pages 77–, New York, NY, USA, 2004. ACM.

[77] Anton Kaplanyan and Carsten Dachsbacher. 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, pages 99–107, New York, NY, USA, 2010. ACM.