

國立交通大學

資訊工程系

碩士論文

用於球諧函數照明架構之
有效取樣與重建入射光強度之方法



Effective Sampling and Reconstruction of Incident Radiance for
Spherical Harmonic Lighting

研究生：孫毓翔

指導教授：莊榮宏 博士

中華民國九十四年七月

用於球諧函數照明架構之 有效取樣與重建入射光強度之方法

研究生：孫毓翔

指導教授：莊榮宏 博士

國立交通大學資訊工程學系

摘要

球諧函數照明為一即時顯像技術，其能夠有效率的顯示物體在受到動態、變化緩慢的遠方光源照明之下的陰影等光影效果。為了讓此架構可以處理由鄰近光源所造成的照明效果，我們提出一個取樣與重建的架構。在之前的方法中，鄰近光源所形成的入射光強度函數，是透過一個前處理程序來均勻散佈取樣位置。這些樣本接著用一個跟距離有關的函數進行內插，以得到物體表面每一個頂點的入射光強度。由於樣本的散佈方式並不隨著入射光強度而變化，加上內插時所有的樣本不論遠近均被考慮進來，入射光強度的局部變化無法被成功重建。我們提出一個動態根據物體幾何以及入射光強度來散佈樣本的方法，這個方法可以讓使用者指定一錯誤容忍值，以便在樣本數量與重建品質之間進行控制。另外，對於每個樣本，我們都給予一個有效範圍，使得入射光強度的局部變化能夠被有效保存，並且提升重建效率。

Effective Sampling and Reconstruction of Incident Radiance for Spherical Harmonic Lighting

Student: Yu-Hsian Sun

Advisor: Dr. Jung-Hong Chuang

Department of Computer Science and Information Engineering
National Chiao Tung University

ABSTRACT

Pre-computed radiance transfer accounting for self-shadowing and interreflection allows objects to be shaded by distant, dynamic, low-frequency lights [SKS02]. In the previous work, incident radiance function is sampled at points uniformly selected by an off-line process, and a radial function is then used to interpolate all samples. If lights are near to objects, or occluding one another, a large number of samples are required. Since interpolation is performed through all samples, locality can be likely to lose. In this thesis, we present a framework of sampling and reconstruction in which objects can be shaded by nearby luminaries. Our work dynamically selects sample points according to object geometries and nearby luminaries. The quality of sampling is controlled by a user-specified error bound. Furthermore, the valid domain of samples can be used as error bounds to enhance the interpolation.

ACKNOWLEDGMENT

First of all, I would like to thank my advisor, Professor Jung-Hong Chuang and Mr. Wang-Yeh Lee for their guidance, support, and inspiration throughout my master's degree. I am grateful to Mr. Shih-Ling Keng and Mr. Ling-Lin Shih for their comments and encouragements. Thanks to my colleagues in CGGM lab: Chih-Chun Chen, Tan-Chi Ho, Min-Sheng Chien, Ren-Hao Jen, Chi-Han Peng, Jong-Hon Lu, Yong-Cheng Chen, Yi-Gin Lin, Roger Hong, Cheng-Li Hou, and Chao-Wei Juan. It is always a great pleasure to study and exchange ideas with everyone in CGGM lab during the past two years. It is good to have my friends Chih-Hao Liang, Chin-Min Lin, Bo-Han Li, and Jin-Jen Huang to stand by my side. Last but not least, I would like to thank my family for their love and support.



Contents

List of Figures	vi
List of Algorithms	viii
List of Tables	ix
1 Introduction	1
1.1 Literature Review	2
1.2 Thesis Overview	3
1.2.1 Problem Statement	3
1.2.2 Contributions	4
1.2.3 Thesis Organization	4
2 Literature Review	6
2.1 Fundamentals of Physically-Based Rendering	6
2.1.1 Spherical Coordinates	6
2.1.2 Projected Area and Solid Angle	7
2.1.3 Radiometry	8
2.1.4 Light Reflection	10
2.1.5 Formulations of Local Reflection Integral	11
2.2 Reflection Integral and Spherical Harmonic Lighting	12
2.2.1 Reflection Integral as a Convolution Process	12
2.2.2 Orthonormal Basis Functions and Frequency Domain	13
2.2.3 Fast Evaluation of Reflection Integral	14
2.2.4 Precomputed Radiance Transfer	17

2.2.5	Limitations of Spherical Harmonic Lighting	18
2.3	Caching Techniques for Rendering	20
3	Sampling and Reconstruction of Incident Radiance	24
3.1	Framework Overview	24
3.2	Preprocessing Stage	25
3.2.1	Precomputing Radiance Transfer Coefficients	26
3.2.2	Bounding Volume Hierarchy Creation	28
3.3	Runtime Stage	30
3.3.1	The BVH and Its Use for Sampling	30
3.3.2	The Oracle Function	32
3.3.3	Hierarchy Traversal	35
3.3.4	Incident Radiance Sampling	36
3.3.5	Efficient Reconstruction using Octree	37
4	Results	40
5	Conclusion and Future Work	49
5.1	Summary	49
5.2	Future Work	50
5.2.1	GPU Acceleration	50
5.2.2	Higher-order Interpolator	50
5.2.3	Oracle Improvement	50
5.2.4	Real-time Rendering Applications	51
	Bibliography	52



List of Figures

2.1	Notations of spherical coordinates.	7
2.2	Three types of BRDF. Left to right: diffuse, specular, glossy [DBB03].	10
2.3	Visualize spherical harmonics of various bands [RH01].	15
2.4	A light probe and its approximation with 9 spherical harmonic coefficients [RH01].	16
2.5	A directional function and its approximation with spherical harmonics. Left to right: the original function, and the approximation with $l = (0, 2, \dots, 10)$. The number of coefficients are 1, 4, ..., 100, respectively [Gre03].	16
2.6	Approximate various BRDF and lighting condition with 25 coefficients [KSS02].	17
2.7	Precomputed Radiance Transfer [SKS02].	17
2.8	Reconstruct $L(x, \omega_i)$ through ICP generated points.	19
2.9	An example of irradiance caching [PH04].	21
3.1	Framework Overview.	26
3.2	BVH construction for the armadillo model.	30
3.3	Approximating an 1D function with discrete samples.	32
3.4	If the blockers and the receiver are near, higher variations in irradiance occurs [Arv94].	34
3.5	Computing locational codes for nodes and vertices in a 2D quadtree.	38
4.1	Lighting configuration in Fig. 4.2.	41
4.2	Comparisons of the armadillo model for our methods, the previous work with ICP, and the ground truth.	42
4.3	Sampling distribution of our methods and the ICP algorithm.	43
4.4	Different lighting configurations for the church model.	43

4.5	Comparisons of the church model for our methods with $\epsilon = 0.007$, $\epsilon = 0.004$, the ICP method with 256 samples, 512 samples, and the ground truth.	46
4.6	Our methods detect the lighting configurations, and the sample numbers under the same error bound are adjusted accordingly. Both lighting configurations are rendered with $\epsilon = 0.001$	47
4.7	Results of different error bounds for the Buddha model.	48



List of Algorithms

3.1	Compute t_{ip} for a vertex p	27
3.2	Generate uniform direction samples over S^2	28
3.3	Recursively Build BVH for O	31
3.4	BVH Traversal	35



List of Tables

3.1	Coordinate system of each cubemap face [Ope99].	37
4.1	Error and timing comparisons of the armadillo model in Fig. 4.2.	41
4.2	Error and timing comparison of the church model in Fig. 4.5.	44
4.3	Sample numbers and costs under different lighting configurations with our method.	45
4.4	Errors and timing statistics for Fig. 4.7.	45



CHAPTER 1

Introduction

With modern graphics hardware, complex lighting and shading effects seen only in traditional global illumination methods are now possible to be rendered in real time. One possible approach is the precomputed radiance transfer [SKS02], which precomputes complex transfer functions including occlusions and interreflections and encodes this function in textures as spherical harmonic coefficients. These methods allow for fast display of static objects with complex lighting conditions, assuming the distant light that is represented by a single environment map. However, several restrictions limit their applicable areas. For example, the object must not be deformed, the lighting environment can be rotated but not be translated, and the spatial relationship between two objects can not be altered. These methods, however, shed some lights on the way to physically-based, high-quality real-time rendering on graphics hardware.

We present a framework of sampling and reconstruction that allows objects to be shaded by nearby luminaries under the framework of precomputed radiance transfer. In the previous work, incident radiance function are reconstructed at sample points uniformly selected by an off-line process and are interpolated using a radial function over all samples. This method generally requires a large number of samples to fully reconstruct the incident radiance function. Furthermore, how many samples are needed is unknown. We propose a hierarchical sampling scheme that selects sample points according to object geometries and nearby luminaries, resulting in dynamically scalable image that has better quality compared to previous methods. We also bound the errors with the valid domain of samples to better the interpolation.

1.1 Literature Review

Real-time rendering has lots of important applications and is currently an active research area. Traditional graphics hardware has many restrictions. For example, light sources are limited to point light sources. The lighting model is limited to the Phong model. As graphics hardware evolves, we are possible to render a more physically-based, realistic images in real time. To shade a point of interest, one must evaluate the local reflectance integral, which is an integration of the incident radiance function $L(x, \omega_i)$ and a bidirectional reflectance distribution function (BRDF) that models surface reflectance properties over the entire incident hemisphere. The BRDF is a known function, but incident radiance is not. We can separate the problem of the evaluation of the local reflection integral into two: one is how to compute $L(x, \omega_i)$ and the other is how to efficiently compute the integral itself.

The incident radiance function $L(x, \omega_i)$ can be decomposed into direct and indirect illuminations. Direct lighting can be obtained through the monte carlo integration. Indirect illumination can be obtained by global illumination algorithms such as path tracing [Kaj86], photon mapping [Jen01], or radiosity. Indirect illumination is usually ignored in rendering algorithms based on traditional graphics hardware due to its high computational costs, but can be computed by performing radiosity simulation or approximated by dynamic calculations of ambient occlusion or some multi-pass rendering algorithms. Simulating the global transportation of light requires global knowledge of the scene, which is expensive to derive. Precomputation makes it feasible to add global illumination effects into real-time rendering applications. Daubert et al. precomputes the interreflection between surface mesostructures and parameterize the result with viewing and lighting direction [DKS⁺03]. By this way the dynamic interreflection on surface can be efficiently evaluated with a few texture lookups. Hao et al. precomputes the dipole approximation integral for subsurface scattering at many lighting directions, and store the result with a reference point scheme to compress the data set [HV04]. Sloan et al. proposed a method that pre-calculates self-transfers such as occlusion and interreflection and stores the transfer function as spherical harmonic coefficients for each vertex [SKS02]. Even if we consider only direct illumination, we still have to evaluate the local reflection integral. A physically accurate approach to compute the integral is ray tracing with monte carlo integration, as proposed by Kajiya [Kaj86] and Cook [CPC84]. However since we must gather radiance values from all directions to shade a point of interest, the coherence between different rays is so low that the

number of rays required may be large. This leads to a slow process and is not inherently suitable for graphics hardware. This is why most graphics hardware supports only a finite number of point light sources. We can intuitively think that evaluating the reflection integral with ray tracing is to compute the convolution of the incident radiance function and the surface BRDF in the directional domain, which is inevitably a slow process. Ramamoorthi and Hanrahan proposed a framework that evaluates the reflection integral in the frequency domain [RH01, RH04] and justified the work done by Sloan and Kautz et al. [SKS02, KSS02]

1.2 Thesis Overview

1.2.1 Problem Statement

In the framework of Sloan et al. [SKS02], distant light sources are assumed and in consequence the radiance field does not vary along surface positions. Incident radiance function $L(x, \omega_i)$ on each point x is represented with only a single directional function $L_p(\omega)$ in spherical harmonics, where p is usually the center of the shaded object. The assumption does not hold true if the light source is not infinite far away, and such violations may result in incorrect shading. It is indicated that the problem posed by non-distant lights could be solved by a multiple sampling of the radiance function across the surface with ICP algorithms, but no justifications is given. Since what we want is a correct shading result, one must take into account both the variation of the incident radiance function $L(x, \omega_i)$ and the transfer function $T(x, \omega)$ to decide the number and distribution of the samples.

We developed a sampling scheme that deals with non-distant light sources. In general, $L(x, \omega_i)$ varies both in positions and directions and can be very complex due to occlusions and complex light sources. As in [SKS02] and [AKDS04], for each sample point x_i in space we can compute its incident radiance function $L_{x_i}(\omega)$ and represent it in a vector of spherical harmonic coefficients. In this framework, each $L(x, \omega_i)$ is point-sampled in discrete points x_i and reconstructed from these sample points. The error introduced by the approximation is determined by two factors: the sampling distribution and the reconstruction filter. The work by Annen et al. [AKDS04] is a higher-order reconstruction filter. Higher-order reconstruction filters can improve shading results, but cannot capture the local properties of the approximated function. In the case of the incident radiance function, changes due to occlusions and complex lighting geometries cannot be captured by higher-order reconstruction filters. To capture the local prop-

erties of the function, more samples must be placed at proper locations. For example, more samples should be placed on the boundary of shadows, as in the work by Ward et al. [WRC88]. We want to address the problem of sample distribution and reconstruction. Our framework dynamically decides the sample number and distribution, accounting for the variation of $L(x, \omega)$. Since it is possible that the variation of $L(x, \omega)$ is too high to be reconstructed in desired computational cost, the framework are able to adjust the sample number, given user-specified error bounds or computational costs.

To dynamically distribute samples and to decide the number of samples needed, a bounding volume hierarchy of the object is created as an off-line process. During run time, we evaluate an estimated error when the incident radiance function for each bounding volume hierarchy node is being sampled. If the estimated error is larger than a user-specified error bound, more samples are added and the positions of samples are determined by the hierarchy. The incident radiance function is reconstructed from samples with the same interpolator used in Sloan et al's work [SKS02, AKDS04]. However, to achieve more efficient reconstruction and to capture local variations of lighting, a valid domain is associated with each sample, and only valid samples are used for interpolation. Moreover, the samples are inserted to an octree for efficient queries, when they are generated.



1.2.2 Contributions

Our contributions can be concluded as follows:

- A hierarchical object-space sampling scheme for spherical harmonic lighting that supports non-distant lights.
- A spatial partitioning data structure that accelerates the reconstruction of incident radiance functions.
- A mechanism that allows users to control the trade-off between the shading quality and the rendering cost.

1.2.3 Thesis Organization

In Chapter 2, we introduce notations and the backgrounds required for our work. In Chapter 3, we describe the framework and theoretical details of our work, while Chapter 4 demonstrates

the results, which are compared to previous methods, and in Chapter 5, we discuss possible improvements and the future work.



Literature Review

This chapter lays the groundwork for physically-based shading on modern graphics hardware. We start from the foundations of physically-based rendering within a traditional realistic image synthesis framework. We then show how spherical harmonics and precomputed radiance transfer allow for fast physically-based shading on modern graphics hardware. Finally, we discuss existing techniques for caching radiance fields and their theoretical backgrounds.

2.1 Fundamentals of Physically-Based Rendering

First we introduce the physical quantities for lighting and rendering. Then we discuss the mathematical formulation that models the interaction between the light and the surface. Finally we discuss how modern graphics hardware is capable of converting simulation results into displayable low-dynamic-range pixels.

2.1.1 Spherical Coordinates

We are interested in quantities about light. Lighting differs from each location and each direction. To define a location in space we use a Cartesian system. In this thesis we assume a right-handed coordinate system. A point in space, usually denoted as p in this thesis, can be explicitly written as an element in R^3 space, (x, y, z) . To denote a vector, we use notations with an arrow above an English alphabet, like \vec{s} . For pure directional quantities, we simply use the term *direction*. In this thesis we denote a direction as ω , and a differential solid angle around a certain direction as $d\omega$. Solid angle will be introduced later. The set of all possible direction, which

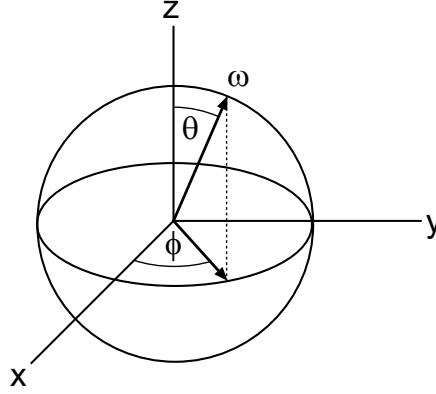


Figure 2.1: Notations of spherical coordinates.

can be imagined as directions from the origin to surface points on a unit sphere, is denoted as S^2 . When calculating a point's outgoing radiance due to reflection, we only get interested the upper hemisphere above the plane formed by the normal at the point of interest. We denote the set of directions of the upper hemisphere as Ω . Though a direction is a unique element in S^2 domain, in practice we often parameterize the directional domain into horizontal angle, ϕ , and vertical angle, θ , as shown in Fig. 2.1. The vertical angle θ is the angle between the direction vector ω and z -axis. The horizontal angle ϕ is the angle between the projection of ω onto the xy plane and the x -axis. Most directional functions, such as spherical harmonics, can be explicitly defined as a function over θ and ϕ . To convert between the angle notation (θ, ϕ) and normalized Euclidean vector notation $\omega = (x, y, z)$ we use the following relations:

$$\begin{aligned} \cos\theta &= z; & \theta &= \cos^{-1}(z); \\ \sin\theta \cos\phi &= x; & \phi &= \tan^{-1}\left(\frac{y}{x}\right); \\ \sin\theta \sin\phi &= y. \end{aligned}$$

2.1.2 Projected Area and Solid Angle

Projected area is defined as the orthogonal projection of a surface of any shape onto a plane with a unit vector v as normal. The differential form is $dA_{proj} = \cos(\theta)dA$, where θ is the angle between the local surface normal and the unit vector v . We can integrate dA_{proj} over the visible surface area to get the total projected area:

$$A_{proj} = \int_A \cos\theta dA.$$

Solid angle is an extension of plane angle from two dimensions to three dimensions. Recall the definition of a plane angle is “One radian is the plane angle between two radii of a circle that cuts off on the circumference an arc equal in length to the radius.” And the definition of a solid angle is extended to: “The solid angle subtended by an object from a point P is the area of the projection of the object onto the unit sphere centered at P [SP94]”. Solid angle is expressed in steradians (sr). The solid angle subtended by the whole sphere is 4π sr, that is, the entire area of a unit sphere. The solid angle of an object is thus the area of the projection of the object onto a unit sphere. Note that two objects different in shape can still subtend the same solid angle. We can think of the differential solid angle as representing both a direction and an infinitesimal area on the unit sphere. We use $d\omega$ to denote the differential solid angle around a certain direction ω . Sometimes given a point P in space, we need to convert between differential solid angle and differential surface area. Let r be the distance between P and the differential area dA , and θ is the angle between normal of dA and the vector from dA to P , the differential solid angle subtended by dA is:

$$d\omega = \frac{\cos\theta dA}{r^2}. \quad (2.1)$$

If dA_s is a small differential surface element on a sphere of radius r , then

$$dA_s = r^2 \sin\theta d\theta d\phi,$$

and

$$d\omega = \frac{dA_s}{r^2} = \sin\theta d\theta d\phi.$$

2.1.3 Radiometry

Physically-based rendering is the simulation of light. The basic terminology to describe light is radiometry, a measurement of optical radiation, which is electromagnetic radiation within the frequency range between $3 \cdot 10^{11}$ and $3 \cdot 10^{16}$ Hz. This range corresponds to wavelength between 0.01 and 1000 micrometers (μm) and includes the regions commonly called the ultraviolet, the visible, and the infrared.

The most basic quantity in radiometry is the photon. The energy e_λ of a photon with wavelength λ is

$$e_\lambda = \frac{hc}{\lambda},$$

where $h \approx 6.63 \cdot 10^{-34} J \cdot sec$ is Planck's constant, and c is the speed of light. e_λ is measured in joules (J).

Spectral radiant energy Q_λ in n_λ photons with wavelength λ is

$$Q_\lambda = n_\lambda e_\lambda = n_\lambda \frac{hc}{\lambda}.$$

Radiant energy Q is the energy computed by integrating the spectral radiant energy over all possible wavelengths:

$$Q = \int_0^{\text{inf}} Q_\lambda d\lambda.$$

Radiant flux or radiant power Φ describes the flow of radiant energy per unit time:

$$\Phi = \frac{dQ}{dt},$$

where t is measured in second. Radiant flux is often just called flux.

Radiant flux density M or B or E is defined as the differential flux per differential area at certain surface location x :

$$M(x) = B(x) = E(x) = \frac{d\Phi}{dA},$$

where M is referred to as radiant existence, which is the flux leaving a surface, B is referred to as radiosity, which is exactly the same as radiant existence; and E is referred to as irradiance, which is the flux arriving at a surface.

Radiant intensity I is defined as the differential flux per differential solid angle $d\omega$:

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

The essential measurement used in the context of rendering is *radiance*, L , defined as the differential flux per differential projected area per differential solid angle:

$$L = \frac{d^2\Phi}{\cos\theta dA d\omega} = \frac{dE}{\cos\theta d\omega}.$$

Radiance can be described by a five-dimensional function (three for the position and two for the direction) per wavelength, usually written as $L_\lambda(x, \omega)$. It is the most important quantity in radiometry, since it could most closely represent the color of an object. Most light receivers, such as cameras and the human eye, are sensitive to radiance, while the response curve of these sensors may be different. In practice, we often simplify per-wavelength calculation into tristimulus values, $L_R(x, \omega)$, $L_G(x, \omega)$, and $L_B(x, \omega)$. In this thesis, we simply use $L(x, \omega_i)$ and did not distinguish between them, though in the implementation we have to process these three values, respectively.

2.1.4 Light Reflection

In this section we introduce the theoretical framework used to model the interaction between light and object surfaces. Radiant energy will be scattered or absorbed by surfaces. We consider only reflection here.

Reflection is a behavior that light enters and leaves the material at the same point on the surface. The amount of energy been reflected can be modeled by a six dimensional function called *bidirectional reflectance distribution function*, or *BRDF*. BRDF is the ratio between irradiance and outgoing radiance, usually denoted as ρ [DBB03]:

$$\rho(x, \omega_i, \omega_o) = \frac{dL(x, \omega_o)}{dE(x, \omega_i)},$$

where x is the 2D position on the surface, ω_o is the differential solid angle around outgoing direction of radiance, ω_i is the differential solid angle around incident direction of irradiance. BRDF becomes a 4D function for a given surface location, and a 2D directional function if we fix both the surface location and outgoing direction. We denote a BRDF with fixed outgoing direction as $\rho_{\omega_o}(x, \omega_i)$ and a BRDF with both outgoing direction and surface location fixed as $\rho_{p\omega_o}(\omega_i)$. BRDF can be roughly categorized into the following three types, as shown in Fig. 2.2:

diffuse Reflect light uniformly over the entire reflecting hemisphere. The BRDF is simply a constant, also known as the Lambert's model.

specular A perfectly mirror surface that reflects incident radiance into mirror directions. The BRDF is a Dirac delta function.

glossy Most surfaces are neither ideally diffuse nor ideally specular and are called glossy surfaces. Their BRDFs are much more complex.

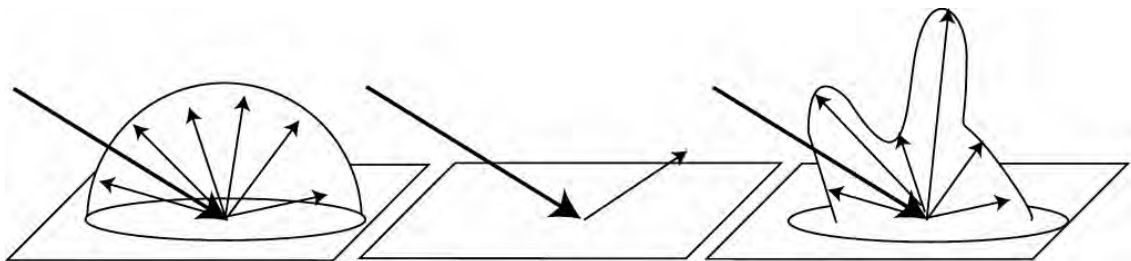


Figure 2.2: Three types of BRDF. Left to right: diffuse, specular, glossy [DBB03].

Given a point x on the surface, the incident radiance function $L(x, \omega_i)$ at the point x , and the BRDF $\rho(x, \omega_i, \omega_o)$, we are interested in computing the outgoing radiance $L(x, \omega_o)$ on a given direction ω_o . We start from the definition of BRDF, integrate over the hemisphere of directions on the right side, finally arriving at the *local reflection integral* [RH04]:

$$\begin{aligned}\rho_\lambda(x, \omega_i, \omega_o) &= \frac{dL(x, \omega_o)}{dE(x, \omega_i)} \\ dL(x, \omega_o) &= \rho_\lambda(x, \omega_i, \omega_o) dE(x, \omega_i) \\ L(x, \omega_o) &= \int_{\Omega_x} \rho_\lambda(x, \omega_i, \omega_o) dE(x, \omega_i) \\ L(x, \omega_o) &= \int_{\Omega_x} \rho_\lambda(x, \omega_i, \omega_o) L(x, \omega_i) \cos(N_x, \omega_i) d\omega, \quad (2.2)\end{aligned}$$

where N_x is the surface normal at the point x , and Ω_x is the hemisphere around the surface normal N_x . The reflected radiance field is given by the integral above. It is an important formula for rendering and will be evaluated at each shading point. As we can see, it is an integration of the incident radiance function and the surface BRDF weighted by the projected area over all possible directions in Ω .

2.1.5 Formulations of Local Reflection Integral

The local reflection integral can be rewritten in various forms. In previous section we introduced a form that integrate over the hemisphere. This is called the *hemispherical formulation*. Another possible form is the *area formulation* [AKDS04]. In this formulation we consider the surfaces of objects in the scene that contribute to the incoming radiance at a point x . We integrate over all surface points on the scene surface A , taking the occlusion and geometry relationship into account. The occlusion can be modeled as a binary visibility function between two points x and y [DBB03]:

$$V(x, y) = \begin{cases} 1 & \text{if } x \text{ and } y \text{ are mutually visible,} \\ 0 & \text{if } x \text{ and } y \text{ are not mutually visible,} \end{cases} \quad \forall x, y \in A.$$

The geometry relationship of two surface points x and y , called the geometry term [DBB03], is:

$$G(x, y) = \frac{\cos(N_x, \Psi) \cos(N_y, \Psi)}{r_{xy}^2},$$

where N_x and N_y are surface normals at point x and y , respectively, and Ψ is the normalized unit vector from x to y . With the geometry term and the visibility function, we can write the

local reflection integral as:

$$L(x, \omega_o) = \int_A L(x, \omega_i) \rho(x, \omega_i, \omega_o) V(x, y) G(x, y) dA_y.$$

The area formulation is useful in computing the lighting from nearby geometries, while the hemispherical form is easier to analyze in the frequency domain. We will have further discussion in the next section.

2.2 Reflection Integral and Spherical Harmonic Lighting

Computing the reflection integral in the frequency domain is an efficient approach. To see why and how this works, we will first show that the reflection integral can be viewed as a convolution in the directional domain. Then we will demonstrate that through orthonormal basis functions, we can project functions defined in the directional domain into the frequency domain, where convolution simply becomes a multiplication. The frequency domain analysis of reflection integral is thoroughly elaborated in the work of Ramamoorthi et al. [RH04].

2.2.1 Reflection Integral as a Convolution Process

The definition of convolution of two functions $f(x)$ and $g(x)$ is given as:

$$f(x) \otimes g(x) = \int f(x)g(t - x) dx.$$

Convolution is usually performed in the entire domains of $f(x)$ and $g(x)$. In the domain of real numbers, we integrate over $[-\infty, \infty]$. In the directional domain, the entire sphere of directions, S^2 , is considered instead. Moreover, we can generalize the notion of convolution to some other transformation R_t [RH04], where R_t is a function of t and write

$$(f(x) \otimes g(x))(t) = \int_x f(x)g(R_t(x)) dx.$$

When R_t is a rotation by angle t , the above formula defines the convolution in the angular domain. In the case of reflection, there is no value on the hemisphere below the surface; hence we integrate only over Ω , the upper hemisphere.

As in [SKS02], we define a *transfer function* $T_{p, \omega_o}(\omega_i)$ for a given point p on surface and a give viewing direction ω_o as the multiplication of the binary visibility function $V_p(\omega_i)$ representing the occlusion at point p , the BRDF $\rho_{p, \omega_o}(\omega_i)$, and the projected solid angle $\cos\theta$. For

diffuse surfaces, the BRDF does not depend on the viewing direction, so we can discard the ω_o term in the transfer function. Note that $V_p(\omega_i)$ is defined as the self occlusion of the object:

$$V_p(\omega_i) = \begin{cases} 1 & \text{if } p \text{ is not occluded in direction } \omega_i, \\ 0 & \text{if } p \text{ is occluded in direction } \omega_i. \end{cases}$$

Now the transfer function for diffuse surfaces is:

$$T_p(\omega_i) = V_p(\omega_i)\rho_p(\omega_i)\cos\theta. \quad (2.3)$$

To see why Equation (2.2) can be seen as a convolution, we can rewrite Equation (2.2) as the multiplication of the incident radiance function and the transfer function:

$$L_p(\omega_o) = \int_{\Omega} L_p(\omega_i)T_p(R_0(\omega_i))d\omega_i \quad (2.4)$$

$$= L_p(\omega_i) \otimes T_p(\omega_i), \quad (2.5)$$

where p is the point of interest and R_0 is an identical rotation.

We can now consider $L_p(\omega_o)$ as a convolution of the incident radiance L_p and the transfer function T_p . Intuitively, a convolution in spatial domain will become a multiplication in frequency domain. To transform L_p and T_p into the frequency domain, we need a set of orthonormal basis functions that span the entire Ω domain.

2.2.2 Orthonormal Basis Functions and Frequency Domain

A pair of functions $f_i(x)$ and $f_j(x)$ are said to be orthonormal to each other if they are normalized and orthogonal to each other. A function $f_i(x)$ is normalized if

$$\int_a^b f_i(x)f_i(x) dx = 1.$$

Two functions $f_i(x)$ and $f_j(x)$ are said to be orthogonal to each other if

$$\int_a^b f_i(x)f_j(x) dx = 0.$$

We may say that if a set of function $\{f_i(x)\}$ are orthonormal to one another, then the convolution of any two functions f_i and f_j in the set is exactly a Kronecker delta function δ_{ij} :

$$\int f_i(x)f_j(x) dx = \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Let $\{B_i(x)\}$ be a complete set of orthonormal functions. Assume that $B_i(x)$ and $f(x)$ have the same range. The function $f(x)$ can be expanded as

$$F_i = \int f(x)B_i(x) dx, \quad (2.6)$$

where

$$f(x) = \sum_i F_i B_i(x). \quad (2.7)$$

We say that F_i are the coefficients obtained by projecting $f(x)$ onto the basis functions $B_i(x)$. After projecting two functions, $f(x)$ and $g(x)$, onto the same set of basis functions, the convolution of $f(x)$ and $g(x)$ will become the sum of products of their respective coefficients [NRH04], that is,

$$\begin{aligned} f(x) \otimes g(x) &= \int f(x)g(x)dx \\ &= \int \left(\sum_i F_i B_i(x) \right) \left(\sum_j G_j B_j(x) \right) dx \\ &= \sum_i \sum_j F_i G_j \int B_i(x) B_j(x) dx \\ &= \sum_i \sum_j F_i G_j \delta_{ij} \\ &= \sum_i F_i G_i. \end{aligned} \quad (2.8)$$

If we project L_p and T_p in Equation (2.2) onto some basis functions over the hemispherical domain, the integral will become simply a dot product between their coefficients. Such a set of functions as $B_i(x)$ is called spherical harmonic functions.

2.2.3 Fast Evaluation of Reflection Integral

Spherical harmonics is used extensively in computational chemistry and introduced to the context of computer graphics by Cabral et al. [CMS87]. They form a complete orthonormal basis set for S^2 domain. Spherical harmonics, denoted as Y_m^l here, are actually the product of terms of Fourier basis functions $\cos\theta e^{im\phi}$, associated Legendre polynomials $P_l^{|m|}$, and a normalization factor K_l^m , as follows:

$$Y_l^m(\theta, \phi) = K_l^m P_l^{|m|}(\cos\theta) e^{im\phi}.$$

The indexing variable $l \geq 0$ and $-l \leq m \leq l$ can be thought as the frequency of the basis function, where m controls the frequency in horizontal directions. Since $-l \leq m \leq l$, we can

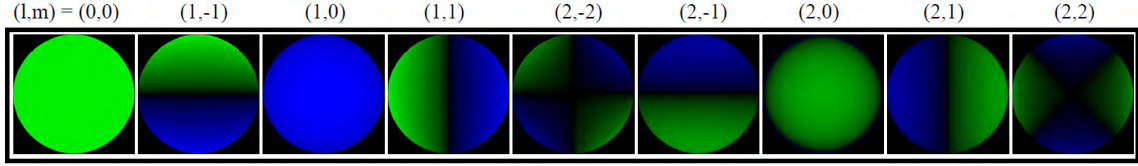


Figure 2.3: Visualize spherical harmonics of various bands [RH01].

combine l and m into a single indexing variable i with $i = l(l + 1) + m$, where $i \geq 0$. Though Y_i forms a complete system only when $i \rightarrow \infty$, most directional functions can be approximated by a small number of basis functions.

To compute the exact value of a spherical harmonic Y_i at certain direction, we must evaluate the associated Legendre polynomial, multiplied by the normalization factor and terms of Fourier basis function. Normalization term is defined as follows [Gre03]:

$$K_m^l = \sqrt{\frac{(2l + 1)(l - |m|)!}{4\pi(l + |m|)!}}$$

$P_l^m(x)$ is computed with the following recursion rules [Gre03]:

$$P_m^m = (-1)^m (2m - 1)!! (1 - x^2)^{m/2} \tag{2.9}$$

$$(l - m)P_m^l = x(2l - 1)P_{l-1}^m - (l + m - l)P_{l-2}^m \tag{2.10}$$

$$P_{m+1}^m = x(2m + 1)P_m^m \tag{2.11}$$

Both Rule (2.10) and (2.11) can be used to compute a higher band (with larger l) solution from a lower band one. Rule (2.11) introduces more floating point roundoff error and Rule (2.10) is used whenever possible. Rule (2.9) is the termination rule of the recurrence relation, and $k!!$ means the product of all odd integers less than or equal to k . Fig. 2.3 visualizes spherical harmonics with $l = (0, 1, 2)$. Since $m \geq -l$ and $m \leq l$, there are nine spherical harmonic functions. Positive values are in green and negative values are in blue. To project a known function defined over all directions, $f(\omega)$, onto spherical harmonic coefficients, one usually evaluates the integral in Equation (2.6) with numerical methods like the quadrature rule. Fig. 2.4 demonstrates projecting an environmental map that represents incident radiance from the surrounding environment onto spherical harmonic coefficients, and reconstruct the map using Equation (2.7). Note that how sharp lighting details are blurred, since only 9 coefficients are used to approximate the environmental map. As the number of coefficients increases, we can preserve more on sharper features. This is shown in Fig. 2.5. At first a small number of coefficients seems to

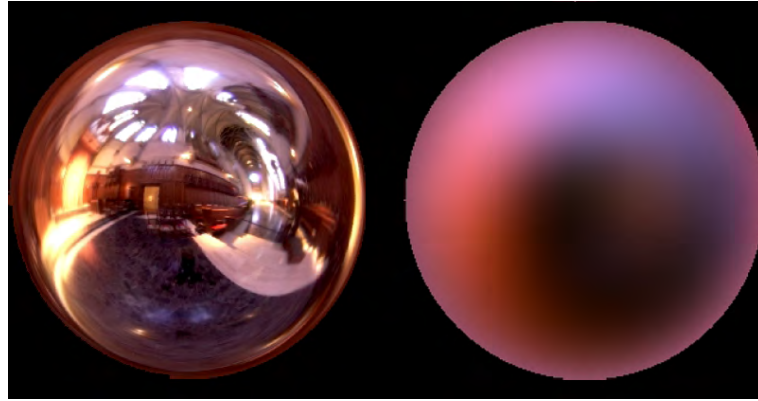


Figure 2.4: A light probe and its approximation with 9 spherical harmonic coefficients [RH01].

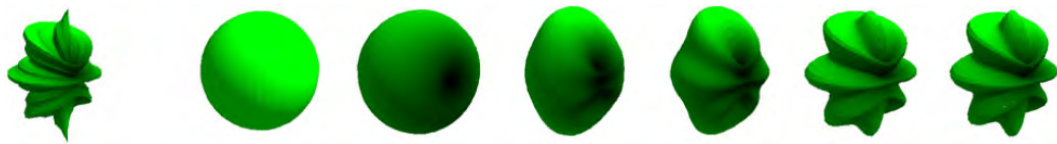


Figure 2.5: A directional function and its approximation with spherical harmonics. Left to right: the original function, and the approximation with $l = (0, 2, \dots, 10)$. The number of coefficients are 1, 4, ..., 100, respectively [Gre03].

be a very rough approximation, but taking surface material into consideration, it is often sufficient. In [RH01], it is shown that for Lambertian surfaces, nine coefficients are sufficient. It is proved that for lambertian surfaces, the error introduced by this approximation is negligible in [RH04]. Kautz et al. proposed a method that stores the spherical harmonic coefficients of BRDF $\rho_{\omega_o}(x, \omega_i)$ on a texture [KSS02], parameterized by viewing angle ω_o , to represent arbitrary BRDF. The result is shown in Fig. 2.6. In their work, 25 spherical harmonic coefficients is sufficient to represent surfaces from perfect lambertian to anisotropic glossy metallic one for each viewing direction. However, their method cannot handle specular surfaces.

Now we know that the reflection integral is a convolution of the incident radiance function and the transfer function. We can evaluate the convolution by projecting both functions onto spherical harmonics to obtain two vectors of spherical harmonic coefficients, and the reflected radiance becomes the dot product of two coefficient vectors as Equation (2.8) shows. If we assume all emitters are infinitely far away, then the incident radiance function does not depend on the position. This is an assumption commonly used in environmental-map based rendering techniques. Now the the incident radiance function can be represented by a vector of spherical

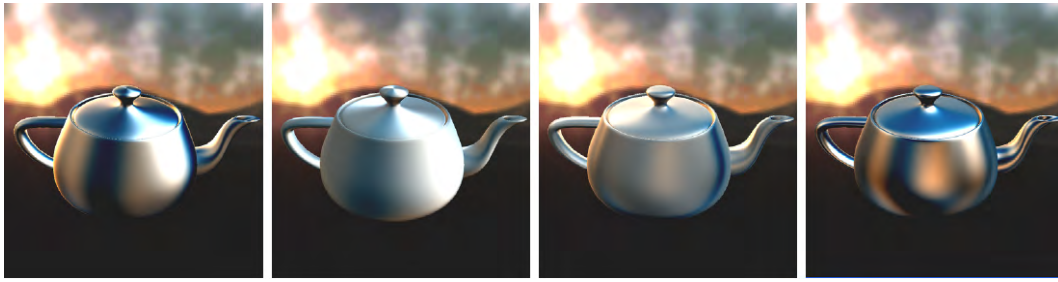


Figure 2.6: Approximate various BRDF and lighting condition with 25 coefficients [KSS02].

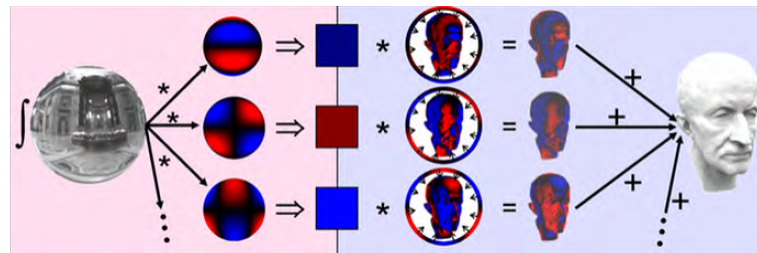


Figure 2.7: Precomputed Radiance Transfer [SKS02].

harmonic coefficients, and can be quickly convoluted. Since the transfer function $T_{\omega_o}(x, \omega_i)$ is different at each shading point, we must precompute them. In the next section, we introduce previous works on precomputing the transfer function.

2.2.4 Precomputed Radiance Transfer

Various representations may encapsulate precomputed or acquired global illumination solutions. Light maps store radiosity simulation results in surface diffuse textures and add global illumination to interactive 3D environments. Surface light fields [CBCG02] record 4D exiting radiance sampled over an object's surface, but both the emitters and the scene must remain static. Horizon map [Max88] stores precomputed visibility information for surface micro-geometry, which can efficiently render self-shadowing effects. Polynomial texture maps [MGW01] fit acquired or precomputed surface BRDF into polynomial functions of incident light vector, allowing for real-time interreflection effects. However, both horizon map and polynomial texture map are limited to point light sources, since for area light sources costly multi-pass rendering is required. As shown in Fig. 2.7, Sloan et al. [SKS02] precompute the transfer function for each vertex, and compute the dot product on graphics hardware, resulting in real-time rendering with effects like occlusions and interreflections on modern graphics hardware. They call T the

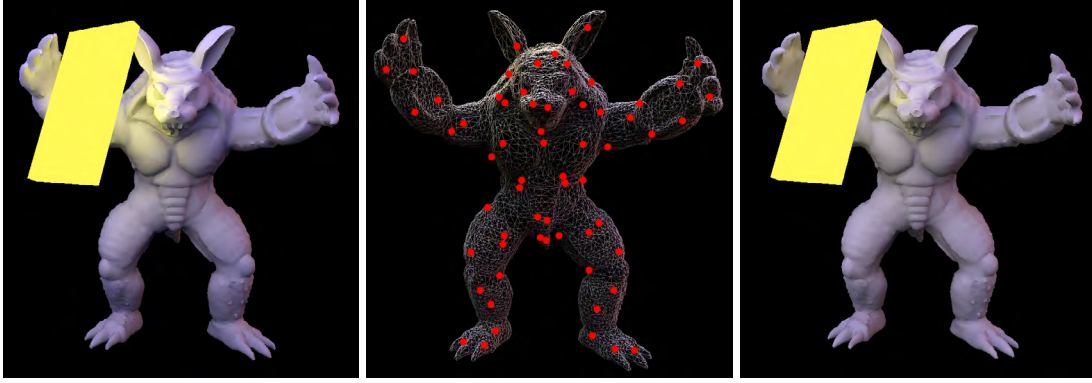
radiance transfer function or simply the transfer function, and we follow their denotation.

The precomputed radiance transfer framework proposed by Sloan et al. [SKS02], also known as *spherical harmonic lighting* [Gre03] that computes Equation (2.2) on frequency domain by projecting both the incident radiance function and the transfer function onto spherical harmonic basis functions. Sloan's work extends the work of Ramamoorthi et al. [RH01], which considers only diffuse surface and precomputes self-shadowing and self-interreflection. Their goal is to shade a object with an environmental map as distant light sources in real time. They also separate surface BRDFs out from the transfer function, thus allowing dynamic changes of surface materials. The framework is further extended by various researchers to address arbitrary BRDFs [KSS02] and compress high-dimensional transfer functions by clustered principal component analysis [SHHS03]. To prevent the transfer function from being recomputed when the surface BRDF is changed, they further decouple the BRDF from the transfer function. By this way the dot product becomes a matrix multiplication. For details, see [SKS02] and [SHHS03]. Ng et al. [NRH04] use haar basis functions defined on directional domain instead of spherical harmonics to handle all-frequency lighting conditions, including specular reflections.

2.2.5 Limitations of Spherical Harmonic Lighting

Though spherical harmonic light provides an efficient way for representing and computing radiance functions, it has its weaknesses and limitations. Spherical harmonics is very inefficient for high-frequency lighting conditions such as point light sources and perfect specular reflections. Hence, the scene must remain static, and only infinitely far environmental light sources can be handled.

Spherical harmonic lighting cannot handle point light sources and specular reflections because of its frequency domain nature. Remember that though $\{Y_i\}$ forms a complete orthonormal basis system, $\{Y_i\}$ is a infinite set; and in practice we can only compute a finite set of coefficients to represent function $f(\omega)$ defined over the directional domain, such as the incident radiance function at some point p . If $f(\omega)$ contains high frequency components, a finite set of coefficients can not be able to accurately represent $f(\omega)$. Point light sources and specular reflections are Dirac delta functions in directional domain, thus requiring an infinitely large number of coefficients. Therefore spherical harmonic lighting is suitable for slow varying, low-frequency lighting conditions. Luckily most natural lighting environments satisfy this condition. To overcome this problem, we can choose another set of basis functions that can represent



(a) Ground truth

(b) Sample location

(c) Result of reconstruction

Figure 2.8: Reconstruct $L(x, \omega_i)$ through ICP generated points.

high-frequency signals more efficiently, as in [NRH04] and [LSSS04].

The second limitation is the nature of precomputation. Since the transfer function are pre-computed over an object, once the object is deformed, the precomputed coefficients are invalid. One possible solution is to encode all possible deformations in advance and use principal component data reduction to relate pose deformation to the change of transfer function coefficients [JF03]. However this solution is only applicable to objects that are not deformed too much—rubber models, for example.

The third limitation, also the problem we'd like to address in this thesis, comes from the fact that we assume that emitters are infinitely far away. If there are emitters near the object being shaded, the incident radiance function $L(x, \omega_i)$ may vary rapidly along x . The previous work [SKS02] simply stated that we may solve this problem through multiple samples over objects and interpolate through linear combinations of coefficients with the following Equation [AKDS04]:

$$L_p = \sum_i w_i L_i, \quad (2.12)$$

where

$$w_i = \frac{1}{\|p-x_i\|^b} \cdot \frac{1}{\sum_j \frac{1}{\|p-x_j\|^b}}. \quad (2.13)$$

In Equation (2.13), x is the location of a sample point, L_i is a coefficient of a sample, and b controls the locality of reconstruction result. Both [SKS02] and [AKDS04] distribute the sample with iterative closest point algorithm(ICP) [LBG80], which is a vector quantization algorithm also known as LBG because of its inventor. This algorithm distributes samples evenly

across all vertices, though the samples are unnecessarily fall on mesh surface. While this may be a good choice if the emitter is not close to the object, it does not give any relationship between the quality, the number of samples, and error bounds. As the emitters draw near to the object or occluding one another, very large number of samples are required to capture the local details of the incident radiance. Since $L(x, \omega_i)$ is reconstructed by interpolating all sample points, reconstruction is slow when there are many samples. The effect of b in Equation (2.13) relies on the distance between samples, which is hard to control and predict. As shown in Fig. 2.8, the armadillo model is illuminated by two distant bluish light sources and one yellow nearby light source. Though 64 samples are used, the lighting near the head of armadillo is missing. Ground truth is obtained by sampling $L(x, \omega_i)$ at every vertex.

Another problem introduced by non-distant light sources is that transfer function stores only self-occlusion or self-interreflection information. If the emitter intrudes the object's convex hull, or there are multiple objects close to one another, the precomputed transfer function becomes invalid. In this thesis we address the problem of varying $L(x, \omega_i)$. The problem of invalid $T(x, \omega_i)$, however, may need further investigation.

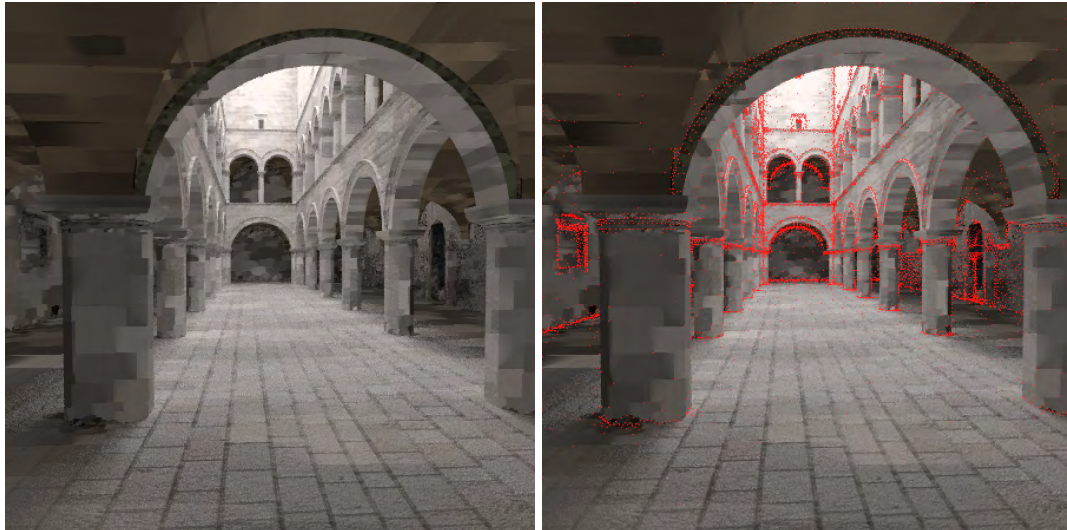
2.3 Caching Techniques for Rendering

Computing the exact value $L(x, \omega_i)$, whether or not indirect illumination is included, is a rather expensive operation. There are several ways to accelerate this operation, including precomputation and sparse sampling. We discuss sparse sampling here.

The idea behind sparse sampling can be considered as a function approximation from discrete sample points. $L(x, \omega)$ is an unknown function, but can be evaluated at certain points (x_i, ω_i) where $x_i \in R^3$ and $\omega_i \in S^2$. We can construct another function $L'(x, \omega)$ from each point (x_i, ω_i) , trying to minimize the difference between L and L' . The difference between two function $f(x)$ and $g(x)$ is usually measured by L_p norm, as defined in [Wat80]:

$$\text{norm}_p(f, g) = \|f - g\|_p = \left(\int |f(x) - g(x)|^p \right)^{-p}.$$

The error introduce by the approximation is related with three factors: the number of samples, the filter used to reconstruct the function, and the distribution of samples. Increasing the sample density or simply the sample number leads to better approximation at the expense of higher sampling costs. If function sampling is expensive, then we should avoid evaluating too many



(a) Scene rendered with irradiance caching

(b) Distribution of caches

Figure 2.9: An example of irradiance caching [PH04].

samples; and proper sample distribution becomes important. After evaluating samples, to construct the approximating function, we must apply some kind of interpolator or filter to existing samples. Possible choices include linear combination and piecewise polynomial approximating functions. In general, high order interpolator gives smooth results, but needs more samples as input and extra computational costs. Several methods exploiting the sparse sampling over spatial domain. We briefly introduce them since they are somehow related to our work. Ward et al. [WRC88] proposed a method that greatly improves the rendering speed for ray tracing diffuse surfaces. The idea is to sparsely sample the incident indirect irradiance function over surfaces. In diffuse environments, irradiance $E_{indirect}(x)$ due to indirect illumination cannot change rapidly, hence justifying the motivation of sparse sampling. To decide the sample distribution, Ward et al. proposed a split sphere model that decides the valid domain of each sample point, which is called *irradiance cache* in their work. The split sphere model has the largest gradient possible for an environment without concentrated sources. During the ray tracing process, at each shading point p (the first hit point of primary ray), they check if there is any usable cache nearby. Nearby usable caches are then used to interpolate the irradiance at p . If there is no usable cache nearby, a new cache is generated at p and stored into an octree constructed from scene geometry for efficient queries. Since each cache generation involves computing irradiance $E(p)$ at p which dominates the computational costs of ray tracing diffuse environments, we can control the quality and speed by scaling the valid domain of each cache. The irradiance

caching is very suitable for rendering scenes containing numerous diffuse objects. Note that cache distribution is decided by the ray tracing order of pixels and the valid domain of each cache, although the caches themselves are stored in R^3 space. As shown in Fig. 2.9, caches are placed densely where indirect illumination changes rapidly, e.g., corners of walls. The blocky artifacts on the pillars and walls due to insufficient sampling can be compensated by computing gradients of irradiance during cache sampling and by using the gradients when interpolating irradiance values [WH92].

In [WRC88], they reconstruct the irradiance $E(x)$ at each point of interest x as the weighted sum of irradiance values E_p of caches [PH04]:

$$E(x) = \frac{\sum_p w_p E_p}{\sum w_p},$$

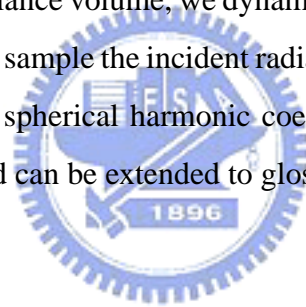
where $w_p = \left(\frac{d_{max}(N_p \cdot N_x) - \|x - p\|}{d_{max}(N_p \cdot N_x)} \right)^2$, and d_{max} is a user-specified upper bound of cache's valid domain.

While the irradiance caches are stored in R^3 space separated from the scene geometry, they represent irradiance of each single point on a surface. These information may become invalid as we move away from the sample position, or the surface normal is rotated. Irradiance volume [GSHG98] generalizes the idea of irradiance caching by storing irradiance values for all possible directions of normals at each sample point (densely sampled for around 200 directions) and arranging sample points in a bilevel uniform grid. Also, irradiance volume stores both direct and indirect irradiance, while irradiance cache stores only indirect irradiance. Greger et al. [GSHG98] define a *semi-dynamic* environment in which most surfaces are stationary, and a few dynamic objects are small relative to the scale of the environment. For example, positioning a piece of furniture in an architectural application would fall into this category. With irradiance volume, not only irradiance on existing surfaces can be interpolated, but also a few dynamic objects moving in this semi-dynamic environment can be shaded by irradiance volume computed from the stationary geometries. However, a few moving objects cannot affect the shading results of the stationary geometries, since the irradiance volume is precomputed and static.

The sample density in an irradiance volume is decided by the number of primitives falling inside a grid cell. If the number of primitives inside a grid cell exceeds some user-specified limit, the cell is further divided into another uniform grid, resulting in a bilevel uniform grid. The sample positions are corner points of grid cells. To shade a point p with normal N_p , one

must query the irradiance volume to find its corresponding irradiance. To query the irradiance at p with normal N_p , the cell containing p is accessed to find the sample values corresponding to the direction N_p . Since the cell has eight corner points, a trilinear interpolation is used to reconstruct the final irradiance from the eight values retrieved from the cell. Unlike Ward et al's work, irradiance volume gives no error bounds for the interpolation.

Both irradiance caching and irradiance volume require a long computational time, and the sampling is done off-line. On the other hand, our work targets interactive 3D environments under the framework of spherical harmonic lighting. Moreover, since irradiance cache and irradiance volume store irradiance rather than directional radiance, they are limited to diffuse environments. Our work adopts the idea of the valid domain from irradiance caching, but use a different sampling strategy. Irradiance caching places the sample points in the progress of ray tracing, whereas our method distributes our samples as the object hierarchy is traversed. Our method has some similarities with irradiance volume as we both place samples in R^3 rather than on object surfaces. Unlike irradiance volume, we dynamically determine the sample distribution and the number of samples; and sample the incident radiance function at runtime. Also, we store directional radiance values (in spherical harmonic coefficients) rather than irradiance values, and in consequence, the method can be extended to glossy materials as in [KSS02].



Sampling and Reconstruction of Incident Radiance

3.1 Framework Overview

For distant lights, the incident radiance function $L(x, \omega_i)$ could not change a lot around x and hence vertices can use the same incident radiance coefficients. For non-distant lights, this is no longer true. Our goal is to incorporate non-distant light sources to the spherical harmonic lighting. To prevent from sampling the incident radiance at each vertex, we use the sparse sampling and from which to reconstruct the spherical harmonic coefficients of $L(p, \omega)$ for a vertex p . Sparse sampling is error-prone, but the error will be decreased when the number of samples increases. The problems are that we have to decide where should the samples be placed and when to stop sampling. To dynamically distribute samples and decide the required number of samples, a bounding volume hierarchy (BVH) of the object is constructed during the preprocessing stage. At runtime we traverse the hierarchy, and at each node of the hierarchy, we sample the incident radiance and compute an estimated error. If the error is lower than a user-specified error bound, we stop traversing the hierarchy and terminate the sampling stage. After stopping traversing the hierarchy, the spherical harmonic coefficient vectors representing the incident radiance of every vertices are then reconstructed from those sparse samples.

Previous methods interpolate all samples for the value at a vertex [SKS02, AKDS04]. If the number of samples is larger than a hundred or more, interpolating through all samples becomes very slow. In our framework, to preserve the locality of lighting and to accelerate

the reconstruction speed, each sample is associated with a valid domain. During reconstructing the coefficient vector for the incident radiance at a vertex, only samples whose valid domain contains the vertex will be considered.

Our framework considers the whole scene geometry as a static object that can be lit by dynamic emitters. Both the object and the emitters are represented by triangle meshes. The scene geometry is denoted as O . Geometry of emitters is denoted as G_l . A emitter can be placed at any position as long as it does not intrude the convex hull of O . The size of an emitter should be large enough, since spherical harmonic lighting prefers area light sources and cannot handle point light sources.

As shown in Fig. 3.1, our framework consists of a preprocessing stage involving O and a runtime stage that involves both O and G_l . The preprocessing stage computes the transfer function for each vertex in O and build an axis-aligned bounding volume hierarchy for O . At runtime the incident radiance function is dynamically sampled and reconstructed. Thus the runtime stage can be further divided into two stages. The first stage is a breadth-first traversal of the BVH aiming to determine the number and the distribution of samples by means of the oracle described in section 3.3.2; and to sample the incident radiance function at the same time. Sampling $L(x_i, \omega)$ at a sample position x_i is performed with hardware cubemaps and is described in Section 3.3.4. The second stage reconstructs the vector of spherical harmonic coefficients \vec{l}_p representing the incident radiance function at every vertex p in O by querying the octree data structure and then evaluates the reflection integral for each vertex using Equation (2.8), which is discussed in Section 3.3.5. Shading is calculated with spherical harmonic lighting. The final displaying is done with graphics hardware.

3.2 Preprocessing Stage

The preprocessing stage takes the scene's geometry O as the input. No prior knowledge of emitters G_l is required. The result is a vector of spherical harmonic coefficients for transfer function at each vertex p in O , and a BVH of O . The BVH is organized as a binary tree flattened to an array. Each BVH node stores information for sampling and error estimation such as an averaged position of vertices, radius of bounding box, etc.

First we explain in Section 3.2.1 how to compute the transfer function and its spherical harmonic coefficient vector for each vertex, then in Section 3.2.2 we show how to build the

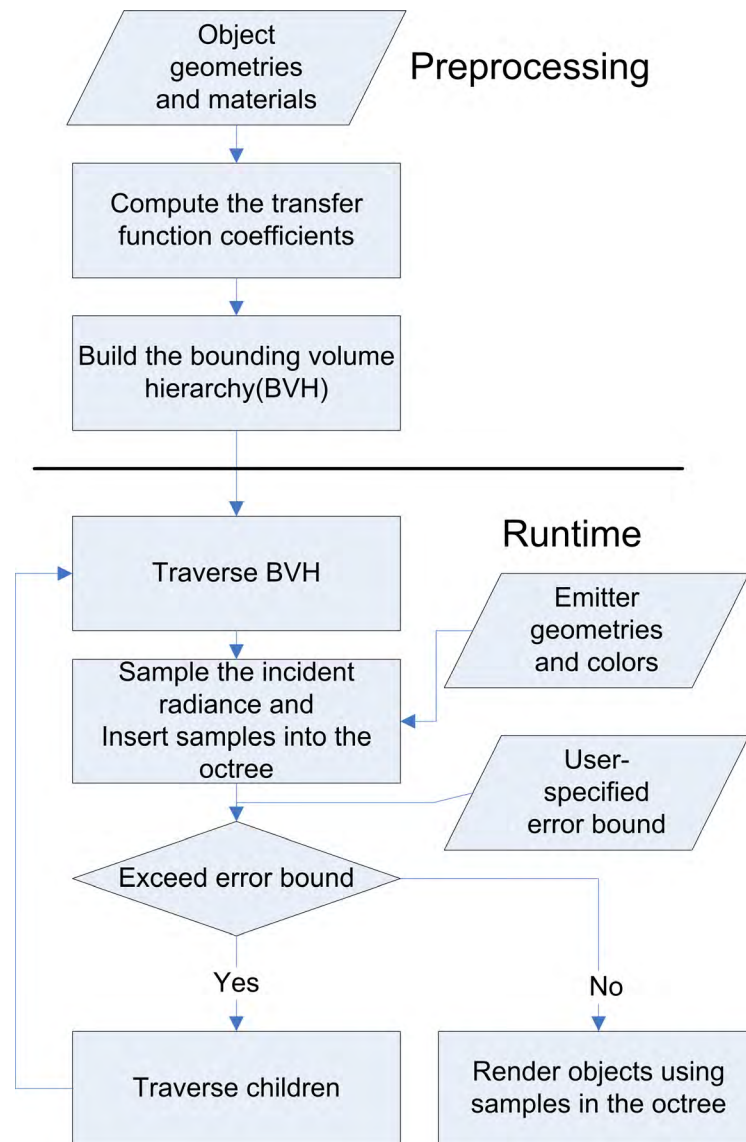


Figure 3.1: Framework Overview.

BVH.

3.2.1 Precomputing Radiance Transfer Coefficients

We precalculate the transfer function $T_p(\omega)$ for each vertex p of O and project $T_p(\omega)$ onto spherical harmonic basis functions as in [SKS02]. The result is a series of spherical harmonic coefficients, denoted as t_{ip} for each spherical harmonic index i and each vertex p . We denote the vector formed by these coefficients as a function $\vec{t}(p)$ defined over all vertices or the function value \vec{t}_p at the vertex p . The transfer function is defined as the product of local surface BRDF ρ , the projected solid angle $\cos\theta$, and the visibility function $V(p, \omega)$ formed by O 's self occlusion.

Algorithm 3.1: Compute t_{ip} for a vertex p

```

input : Vertex  $p$  and its normal  $N_p$  and object geometry  $O$ 
output:  $t_{ip}$  for  $p$ 

1  $\Phi \leftarrow \text{GenerateUniformDirectionSample}(\sqrt{\text{numberOfRay}})$ ;
2 for  $i \leftarrow 0$  to 15 do
3    $t_{ip} \leftarrow 0$ ;
4   foreach  $\omega = (\theta, \phi) \in \Phi$  do
5     convert  $\omega$  into Cartesian coordinate  $\omega'$ ;
6      $\text{ray.origin} \leftarrow p$ ;
7      $\text{ray.direction} \leftarrow \omega'$ ;
8      $\text{vis} \leftarrow \text{IntersectionTest}(O, \text{ray})$ ;
9      $t_{ip} = t_{ip} + Y_i(\omega)(N_p \cdot \omega') \cdot \text{vis}$ ;
10   $t_{ip} = \frac{4t_{ip}\pi\rho_p}{\text{numberOfRay}}$ ;

```

Now Equation (2.2) at each vertex p of O ,

$$Lp(\omega_o) = \int_{\Omega} L_p(\omega) V(p, \omega) \rho(\omega_o, \omega) \cos\theta d\omega,$$

can be rewritten as

$$Lp(\omega_o) = \int_{\Omega} L_p(\omega) T_p(\omega) d\omega,$$

where $T_p(\omega)$ is equal to $V(p, \omega) \rho(\omega_o, \omega) \cos\theta$. Here we assume homogeneous materials across the object, and in our current implementation, the BRDFs are simply lambertian materials, which is a constant value, denoted as ρ_o , over the entire object. Note that the simple lambertian model can be further extended to arbitrary materials as in [KSS02]. The BRDFs should be wavelength dependent, but in our implementation we approximate them by tristimulus channels; so there will be T_{pr} , T_{pg} , T_{pb} , totally 3 functions to be projected on to spherical harmonic basis. Since the lambertian model is simply a constant for each channel, we post-multiply the transfer functions with the material's diffuse reflectivity. We compute $V(p, \omega)$ for each p in O by ray-casting 6400 rays in uniformly sampled directions on S^2 ; see details in algorithm 3.2.

We project the transfer function $T_p(\omega)$ at vertex p onto a set of bands i of spherical harmonic

Algorithm 3.2: Generate uniform direction samples over S^2

input : An integer \sqrt{N} , square root of the number of samples desired.

output: Φ , A set of directions (θ_i, ϕ_i) .

/ ξ_1 and ξ_2 are uniform random variables. */*

1 **for** $i \leftarrow 0$ **to** N **do**

2 $\phi \leftarrow 2\pi\xi_2$;

3 $\theta \leftarrow \arccos(1 - 2\xi_1)$;

4 $\Phi \leftarrow (\theta, \phi) \cup \Phi$;

functions $Y_i(\omega)$ to yield coefficient t_{ip} in band i using the following equation

$$\begin{aligned} t_{ip} &= \int_{S^2} V(p, \omega) \rho_o \cos\theta Y_i(\omega) d\omega \\ &= \frac{4\pi}{N} \sum V(p, \omega) \rho_o \cos\theta Y_i(\omega) \\ &= \frac{4\pi\rho_o}{N} \sum V(p, \omega) \cos\theta Y_i(\omega), \end{aligned}$$

where θ is the angle between the surface normal at p and the ray direction ω . The number of bands effects the accuracy of the reconstructed transfer function \tilde{T}_p from t_{ip} . We use 16 bands ($l = 4$) for the approximation, where each band is represented by an IEEE 32-bit floating point number. Note that all t_{ip} are computed under the same coordinate system; thus no rotation is required. Some previous works compute the transfer functions under the coordinate system formed by the local tangent plane on the surface because the BRDFs are defined over the tangent plane [KSS02]. In such approaches, rotations of spherical harmonic coefficients are required. Our implementation computes the coefficients under the same world coordinate system, and sacrifices some flexibility for simplicity. Algorithm 3.1 shows this process. Intersection tests for computing $V(p, \omega)$ are computed with an axis-aligned binary space partition tree for acceleration [PH04].

3.2.2 Bounding Volume Hierarchy Creation

The bounding volume hierarchy is a hierarchical structure of bounding volumes. Vertices in close spatial proximity are put in clusters and the clusters are enclosed in bounding volume.

There are three possible strategies to build a BVH [Eri05]: top-down, bottom-up, and insertion. Usually a bottom-up or insertion approach may result in a tree that is more balanced

than a top-down approach, but is much slower and complex to construct. We adopt a top-down approach because it is fast, simple, and less error-prone. The degree of children is also an issue. Higher degrees decreases the maximum depth of the hierarchy, but in the mean time increase the cost of hierarchy traversal. As stated in [Eri05], we find that the binary tree structure is a reasonable choice.

For example, since in practice our vertices number will not exceed 100000, the depth of a full binary tree will never exceed 18. Moreover, a binary tree can be flattened into an array without pointers for indexing; that is, for a node n_i with array index i , the children of n_i are n_{2i+1} and n_{2i+2} . This is much more cache-friendly, since during a breadth-first search, children with the same parent node will be placed in nearby memory addresses.

The hierarchy is built in a top-down approach. We start from the bounding box of all vertices in O and build an axis-aligned bounding box for the entire object. We then find the longest axis of the bounding box and split the node into two children. The split point is the vertices' coordinate median on the longest axis. This involves sorting the vertices along the longest axis. Splitting with vertices' coordinate median gives us a balanced and near full binary tree, but not necessarily a BVH with optimal bounding volume [Eri05]. However, it still tends to produce a hierarchy with vertex clusters that are near to one another because a typical mesh for spherical harmonic lighting is finely and evenly tessellated. Since the vertices are distributed evenly over the object, split according to their coordinate median tends to minimize the bounding volume of nodes. Algorithm 3.3 shows this process. Fig. 3.2 shows the construction process of the BVH for the armadillo model. Note that some bounding boxes are quite large, since our strategy does not guarantee optimum sizes of bounding volumes. In our experiments, however, those large bounding boxes do not lead to visual artifacts.

Two attributes associated with each node are also precomputed during the creation of the hierarchy. One is the radius of the bounding volume of each node, which is a variable used in evaluating the oracle function to be discussed in section 3.3.2. The other is the sample position for the node, which is the position average of vertices inside the node. Since our bounding primitive is axis-aligned bounding box, the radius of the bounding volume for each node is set to be half of the diagonal length of the AABB.

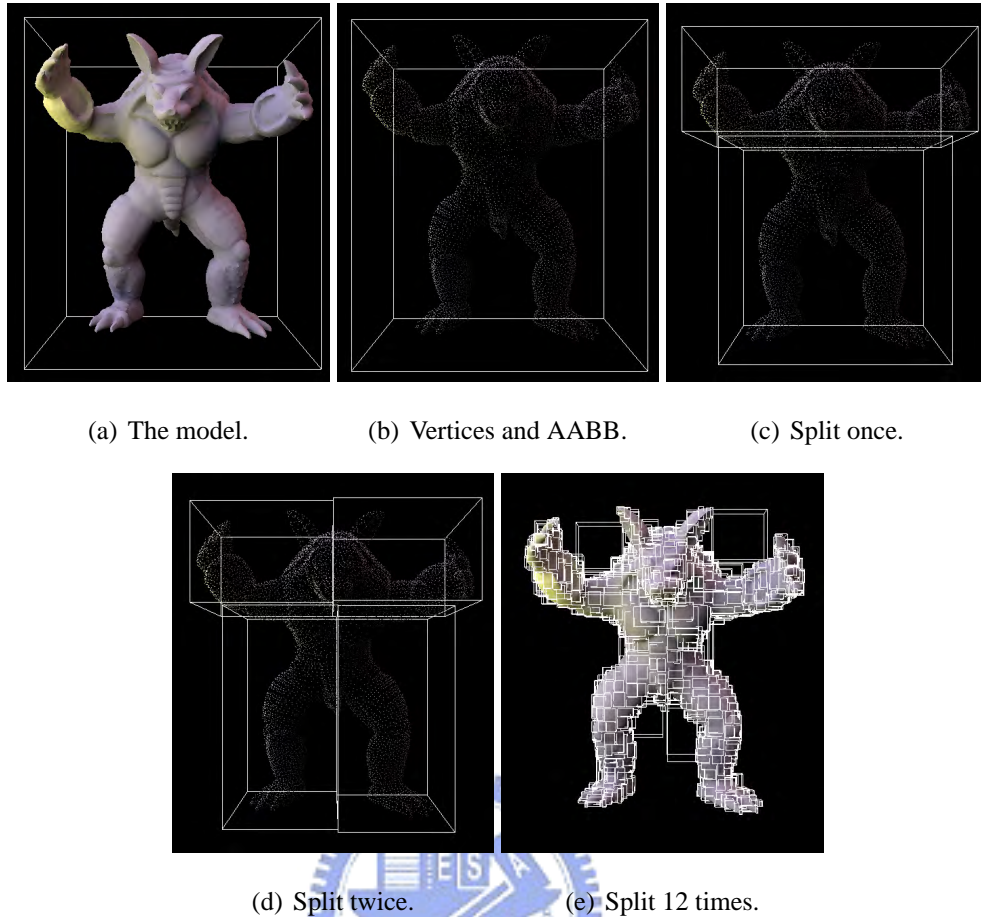


Figure 3.2: BVH construction for the armadillo model.

3.3 Runtime Stage

In this section we first explain how the bounding volume hierarchy and the error estimator are used to distribute samples, and then we discuss how to sample the incident radiance in Section 3.3.4. Finally how the octree is used for efficient reconstruction is described in Section 3.3.5.

3.3.1 The BVH and Its Use for Sampling

For a set of vertices P , the result of shading is the exiting radiance from a vertex $p \in P$ to the camera. The color $C(p)$ of the vertex p is the exiting radiance, computed a dot product of the two coefficient vectors for the incident radiance function and the transfer function, respectively, as shown in Equation (2.8). During run time, the incident radiance coefficient vector \vec{l}_p is unknown, while the coefficient vector for the transfer function \vec{t}_p is precomputed. If we replace

Algorithm 3.3: Recursively Build BVH for O **input** : A BVH node and vertices V inside the node**output**: A binary axis-aligned bounding box tree BVH

```

1 node.AABB ← ComputeAABB( $V$ );
2 node.avgPos ← AveragePosition( $V$ );
3 node.d ← DiagonalLength(node.AABB);
4 laxis ← GetLongestAxis(node.AABB);
5 ( $V_1, V_2$ ) ← SplitAlongMedian( $V$ , laxis);
6 BuildBVH( $V_1$ , node.leftChild);
7 BuildBVH( $V_2$ , node.rightChild);

```

the true coefficient vector \vec{l}_p of the incident radiance function at every vertex p with a vector of coefficients \vec{l}_x sampled at some point x , the error introduced by such approximation $\tilde{C}(p)$ is

$$\|C(p) - \tilde{C}(p)\| = \|(\vec{t}(p) \cdot \vec{l}(p)) - (\vec{t}(p) \cdot \vec{l}_x)\|,$$

where the function $\vec{t}(p)$ is the vector function representing the coefficient vectors for the transfer function at $p \in P$, and in particular with L_1 norm, we have

$$\|C(p) - \tilde{C}(p)\|_1 = \sum_{p \in P} \left| (\vec{t}_p \cdot \vec{l}_p) - (\vec{t}_p \cdot \vec{l}_x) \right|, \quad (3.1)$$

As shown in Fig. 3.3 for 1D cases, if we partition the domain P , which is R^3 in our case and add more samples to the approximation, we get more accurate results. Previous works uniformly partition the domain using the ICP algorithm [SKS02] or uniform grid [GSHG98], here we can do it better. Since the error $\|C(p) - \tilde{C}(p)\|_1$ is defined on vertices rather than the entire R^3 , we can partition the set of all vertices P rather than uniformly partition R^3 . A useful partition on P is the bounding volume hierarchy.

We cannot apply Equation (3.1) to estimate the error of each hierarchy node, since it still involves computing $\vec{l}(p)$ at every vertex. Since it is impossible to know the exact value of $\vec{l}(p)$ unless all vertices p are sampled, our method tries to estimate them. The oracle can be seen as a partition strategy. We don't care about the exact values of error. Only the relative errors between different hierarchy nodes need to be computed. With relative errors, our method decides whether or not a node should be partitioned.

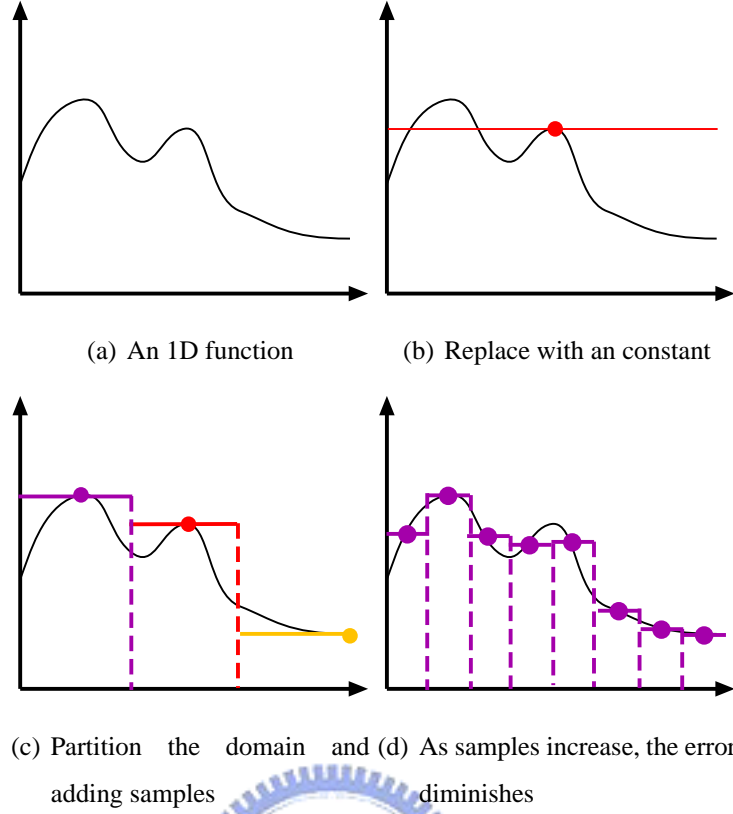


Figure 3.3: Approximating an 1D function with discrete samples.

3.3.2 The Oracle Function

Since the transfer function coefficients $\vec{l}(p)$ is the combination of the visibility function and the surface BRDF, the outgoing radiance will never be larger than the incident radiance, as a result of the energy conservation law. The errors in Equation (3.1) are thus never greater than $\sum_P \left| |\vec{l}_p| - |\vec{l}_x| \right|$, which is further bounded by a looser bound as follows:

$$\max_{p \in P} (|\vec{l}_p|) \|P\|, \quad (3.2)$$

where $\|P\|$ is the number of vertices of the set P . The intuition behind Equation (3.2) is that we will partition a vertex set that has a higher value of $|\vec{l}(p)|$, since discrete sampling introduces larger errors at this region. Note that the norm of vector of spherical harmonic coefficients representing the incident radiance is less than or equal to \sqrt{i} times the L_2 norm of the incident radiance function. The magnitude of the incident radiance function at the sample point x can be obtained by computing the L_2 norm of the incident radiance function:

$$\|L_x(\omega)\|_2 = \sqrt{\int_{S^2} L_x(\omega)^2 d\omega},$$

and the norm of the vector $|\vec{l}_x|$ is

$$|\vec{l}_x| = \sqrt{\sum_i \left(\int_{S^2} L_x(\omega) Y_i(\omega) d\omega \right)^2}.$$

By applying the Cauchy inequality to the norm of the vector $|\vec{l}_x|$, we can see that it is related to the norm of the incident radiance function and have

$$\begin{aligned} |\vec{l}_x|^2 &= \sum_i \left(\int_{S^2} L_x(\omega) Y_i(\omega) d\omega \right)^2 \\ &\leq \sum_i \left(\int_{S^2} L_x(\omega)^2 d\omega \right) \left(\int_{S^2} Y_i(\omega)^2 d\omega \right) \\ &= i \left(\int_{S^2} L_x(\omega)^2 d\omega \right) \\ &= i \|L_x(\omega)\|_2^2, \end{aligned}$$

which implies

$$|\vec{l}_x| \leq \sqrt{i} \|L_x(\omega)\|_2.$$

We can thus use the magnitude of the vector \vec{l}_x to adequately estimate the magnitude of the incident radiance function at point x . In addition to partitioning at brighter regions, vertex set P having a large number of vertices should also be partitioned first, since the error could be more appealing.

However, we must perform sampling to know $|\vec{l}(p)|$ since $\vec{l}(p)$ is known only at run time. As a consequence, our oracle function is an *a posteriori* error estimator; that is, the error estimation is done *after* $|\vec{l}(p)|$ is sampled. We made a few assumptions to further approximate the error.

One assumption is that $\vec{l}(p)$ does not vary too much around p so that we can take a single sample x inside set P and use it to approximate $\max_{p \in P} (|\vec{l}(p)|)$. Another assumption is that the points are evenly distributed on the object surface; so we can use the radius of the bounding volume of P to approximate $\|P\|$. Equation (3.2) can now be approximated by:

$$\max_{p \in P} (|\vec{l}_p|) \|P\| \approx r |\vec{l}_x|,$$

where r is the radius of the bounding volume of P .

Furthermore, we take the possibility of variations into account. The error is higher if there are more variations in $\vec{l}(p)$. We approximate the variation with the ratio of the radius of the bounding volume to the average distance to emitters. The intuition behind this choice is shown

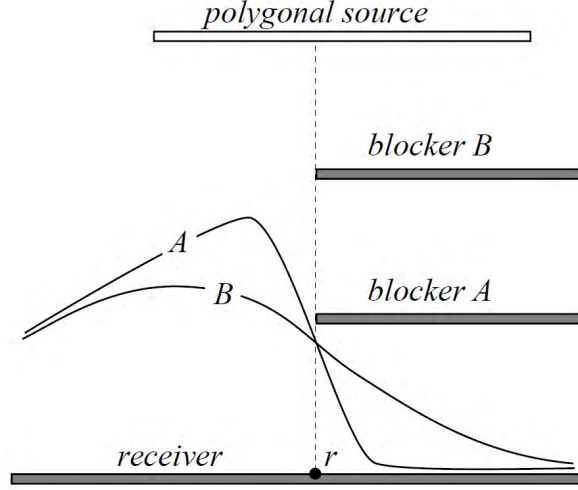


Figure 3.4: If the blockers and the receiver are near, higher variations in irradiance occurs [Arv94].

in Fig. 3.4. If the emitters are far away from our receiver (object), the variation due to nearby blockers could be smaller, as the *B* curve in Fig. 3.4 shows. The blockers can be considered as a type of emitters with less emission. Also, the errors due to the variation are smaller if the receiver is very small compared to the distance to the emitters. One extreme case is a scene lit by infinitely far emitters; thereby no irradiance variations can occur. In the oracle, the variations are approximated by the ratio of the radius of the bounding box of the vertex set P to the harmonic mean of distances from the sample points to all emitters. The harmonic mean of a set of distances d_i , where $i = 1 \dots N$, is [PH04],

$$H_d = \frac{N}{\sum_N \frac{1}{d_i}}.$$

By using the harmonic mean we are able to detect the presence of nearby emitters, as a larger distance has a smaller effect on the harmonic mean. Now we can write our oracle as

$$\frac{r |\vec{l}_x|}{H_d^2}, \quad (3.3)$$

where \vec{l}_x is the vector of coefficients representing the incident radiance sampled at point x , r is the radius of the bounding box of the vertex set P computed in the preprocessing stage, and H_d is the harmonic mean of distances from x to emitters. The computation of \vec{l}_x and H_d will be described in section 3.3.

The oracle function in Equation (3.3) computes the estimated error between $\tilde{C}(p)$ and $C(p)$. Once the estimated error exceeds a user-specified bound, we must traverse to the next level and

Algorithm 3.4: BVH Traversal

```

input: Desired error bound  $\varepsilon$  and sample limit  $N$ 
1 Insert root node into queue ;
2 while queue not empty and  $n_s \leq N$  do
3    $n_s \leftarrow n_s + 1$ ;
4   node  $\leftarrow$  Pop(queue);
5   sample  $\leftarrow$  SampleIncidentRadiance(node.avgPos);
6   InsertIntoOctree(sample);
7    $\epsilon \leftarrow$  Oracle(sample, node);
8   if  $\epsilon > \varepsilon$  then
9     Push(node.left, queue);
10    Push(node.right, queue);

```

evaluate the samples at two children nodes.

3.3.3 Hierarchy Traversal

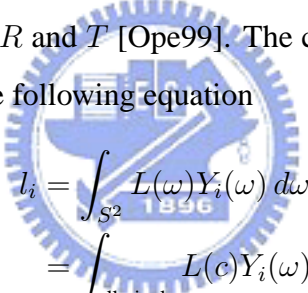
In order to determine the number and the distribution of samples, the bounding volume hierarchy is traversed; and samples are then positioned at proper nodes. Starting from the root, we compute the incident radiance at the sample position precomputed in the preprocessing stage. The oracle function is then evaluated to check whether the sample is adequate for reconstruction. If the oracle exceeds the user-specified error bound, two children of this node are recursively traversed in order to distribute more samples; otherwise, the recursive traversal terminates. Shown in algorithm 3.4, this is performed with a queue. The traversal stops when the oracles evaluated at all nodes being traversed are below the error bound or the number of samples exceeds some user-specified limit. During the traversal, the samples are inserted into an octree for efficient reconstruction, which will be discussed in section 3.3.5. Each sample is associated with a valid domain, aiming to prevent the global reconstruction results from being influenced by local, high-intensity samples. The valid domain is defined as a sphere with radius r centered at the sample's position.

3.3.4 Incident Radiance Sampling

To dynamically sample the incident radiance function at a sample point x , we exploit the use of graphics hardware. We also compute the averaged distance from x to emitters for error estimation at the same time. Emitters G_l are sent down to graphics pipeline six times to render a cubemap. The cubemap now represents the emission of emitters to x . We then project the incident radiance represented by the cubemap onto the spherical harmonic bases. Because cubemapping is formed by perspective projections onto six planes, during the integration we must take into account the non-uniform solid angle associated with each pixel. Let $\omega(c)$ be the solid angle of a pixel $c(s, t)$ of a $k \times k$ cubemap face with viewing direction R . The relation of the cubemap faces and viewing directions R and their corresponding up vectors T is shown in Table 3.1. The vector \vec{c} from the sample point x to the pixel $c(s, t)$ is

$$\vec{c} = \left(\frac{s}{2k} - 1 + \frac{1}{k}\right)S + \left(\frac{t}{2k} - 1 + \frac{1}{k}\right)T + R,$$

where S is the cross product of R and T [Ope99]. The coefficients l_i representing the incident radiance at x is computed by the following equation



$$\begin{aligned} l_i &= \int_{S^2} L(\omega) Y_i(\omega) d\omega \\ &= \int_{\text{all pixels}} L(c) Y_i(\omega) d\omega_c \\ &\approx \sum_{\text{all pixels}} L(c) Y_i(\omega_c) \omega(c), \end{aligned}$$

where $\omega(c)$ is the solid angle associated with pixel $c(s, t)$. We use Equation (2.1) to compute the solid angle $\omega(c)$ of a pixel c from the area of the pixel:

$$\omega(c) = \sum_{\text{all pixels}} L(c) Y_i(\omega_c) \frac{4}{k^2} \frac{\omega_c \cdot R}{|\vec{c}|^2},$$

where $\frac{4}{k^2}$ is the area of each pixel. Since each $k \times k$ cubemap face ranges from -1 to 1 , so $\frac{4}{k^2}(\omega_p \cdot R)$ is the projected area of pixel p .

The sample position is the average position of all vertices inside the node and is precomputed. The number of coefficients for each vector of coefficients \vec{l}_x for sample x is also 16. After the sampling and coefficient computation are done, the vector of coefficients is then used to estimate the error of the node. To estimate the error, we also need the average distance from

R	up vector (T)
$+X$	$-Y$
$-X$	$-Y$
$+Y$	$+Y$
$-Y$	$-Y$
$+Z$	$-Y$
$-Z$	$-Y$

Table 3.1: Coordinate system of each cubemap face [Ope99].

the sample to all the surrounding emitters. As we render the emitters into the cubemap, we also render their distances to the sample into another color buffer. In our implementation this is done with OpenGL's multiple draw buffers and shaders. Then we read the color buffer back to main memory and compute the harmonic mean of distances.

3.3.5 Efficient Reconstruction using Octree

To approximate true incident radiance coefficient vector \vec{l}_p from a given set of samples $\{\vec{l}_x\}$, we first need to find all samples $\{\vec{l}_x\} \in \{\vec{l}_x\}$ whose valid domain contain p and then interpolate \vec{l}_p from these valid samples $\{\vec{l}_x\}$ using method used in [SKS02] and [AKDS04].

The major computation for this step is to query and find all samples whose valid domain contain p . To accelerate such query, a *linear octree* [Eri05] is used. The samples are inserted to this data structure as soon as they are generated during the sampling stage. Different from the pointer-based octree, the linear octree uses a hash table. Each node at a certain level inside the octree has an unique hash key value. The key value depends on both the node's level and position. They are called locational codes [Eri05]. The locational codes in bits are actually the traversal order of the tree encoded in binary bits. Fig. 3.5 shows the locational codes of a quadtree. We can see that locational codes are arranged in the form of $(0.Y_1X_1Y_2X_2Y_3X_3\dots)$. Deeper levels have longer binary fractions.

In our implementation, the binary fractions are represented by integer values. For example, a binary fraction $[0.1100]$ is converted to integer value $[11100]$. The extra bit is called the sentinel bit. The purpose of the sentinel bit is to distinguish from different keys from different level. For example, $[0.00]$ and $[0.0000]$ denote two keys at level one and level two. Their integer value are

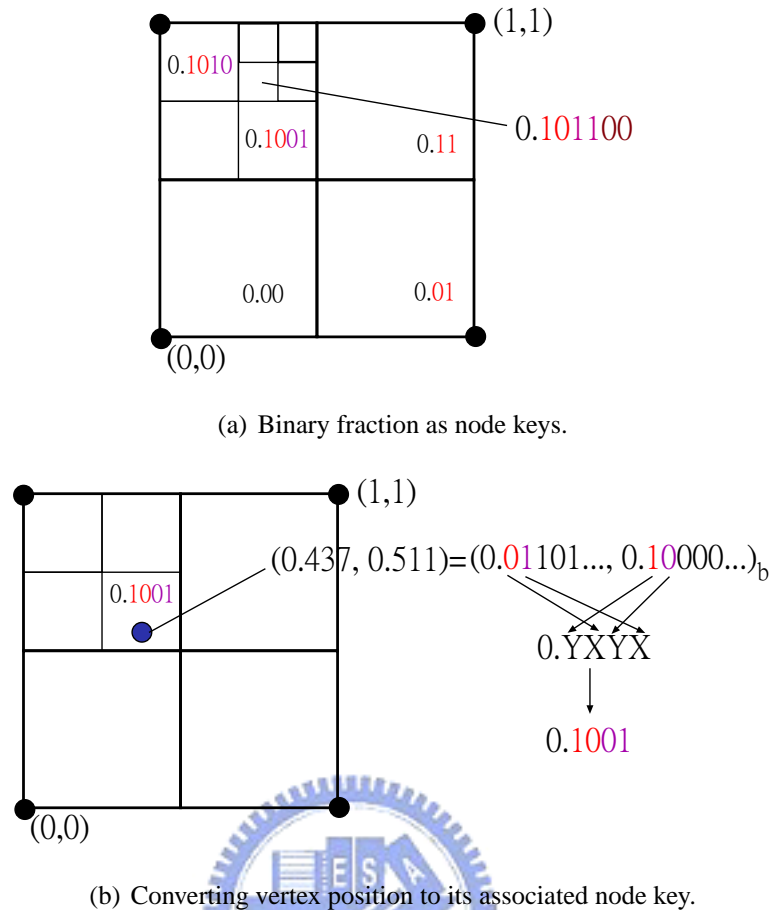


Figure 3.5: Computing locational codes for nodes and vertices in a 2D quadtree.

[100] and [10000], respectively. Therefore given a node with key value k , we can easily find the key value k' of its parent using $k' = k \gg 3$, where \gg is the bitwise right shift operator. For example, the node with integer key value [110] in Fig. 3.5(a) has four children with integer key values [11000], [11001], [11010], and [11011], respectively.

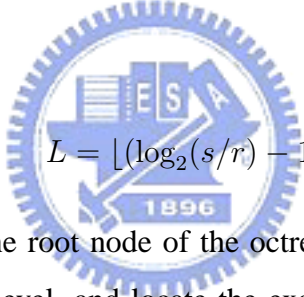
Furthermore, the locational code of the node containing a certain point can be efficiently computed from the coordinates of the point. As in Fig. 3.5(b), the coordinates of the point is normalized to the side length of the root node, ranging from 0 to 1. Converting the coordinates into binary fractions and interleaving their respective fraction bits in YX order yields the locational code of the node containing the point. The locational codes of an octree can be computed in a similar way.

We access each node with a hash table lookup, which is relatively fast compared to traversing a series of pointers. We use a 32-bit integer to define a key, thereby allowing an octree with a max depth of ten levels, which is sufficient in our experiments. The hash table is implemented with the C++ standard template library. For a given sample with a valid domain of radius r ,

we will insert it into a certain level of the octree, whose side length is greater than $2r$ and less than $4r$. By placing in a node with side length greater than $2r$, the sample will be inserted to eight nodes at most. By limiting the node's side length less than $4r$, we can prevent samples with small valid domains being accessed by too many vertices, since nodes with larger side length will be queried more frequently. The samples with the same locational code are stored in a linked list in each octree node. Since each level of the octree node halves its side length, we can directly compute the level L , where a sample with a valid domain of radius r should be inserted, as

$$\begin{aligned} 2r &\leq \frac{s}{2^L} \leq 4r \\ 2^{L+1} &\leq \frac{s}{r} \leq 2^{L+2} \\ \log_2 2^{L+1} &\leq \log_2 \frac{s}{r} \leq \log_2 2^{L+2} \\ L &\leq \left(\log_2 \frac{s}{r} \right) - 1 \leq L + 1, \end{aligned}$$

which implies



$$L = \lfloor (\log_2(s/r) - 1) \rfloor,$$

where s is the side length of the root node of the octree. For any given vertex, we compute the key of the vertex for each level, and locate the exact node in the respective level to find the associated linked list of samples, thus requiring ten hash table lookups, without explicit pointer traversal. Only samples whose valid domain do contain the vertex are considered. We then interpolate through all valid samples by Equation (2.13), but without the power term b for simplicity and speed. Intuitively we can think that nearby samples could have higher weights.

The shading of O is straightforward from now on. For each vertex p , its outgoing radiance is computed by $\vec{t}_p \cdot \vec{l}_p$, the dot product of its precomputed transfer coefficients and reconstructed incident radiance coefficients. In our implementation the dot product is performed by the CPU. The results are then written to an OpenGL *vertex buffer object* binded as a color array for the final rendering. The transformation and rasterization are handled by the graphics hardware.

CHAPTER 4

Results

In this chapter we present several results from our experiments. All images are rendered using the OpenGL API. The hardware platform is a Windows PC with AMD Athlon64 3000+ processor and 512MBytes main memory. The graphics card is an nVidia GeForce6800 with the 8x AGP interface. To compare with the previous method, we implemented Sloan et al.'s diffuse, self-shadowing radiance transfer method [SKS02]. We also implemented the iterative closest point (ICP) algorithm [LBG80] mentioned in their work to find sample locations and to interpolate the incident radiance function for every vertex using Equation (2.13). We compare the visual appearances and the color root mean square error (RMSE) of our sampling and reconstruction methods to those of both the original method in [SKS02] and the ground truth image. The ground truth is obtained by sample the incident radiance function at every vertex. Then the scalability of our methods are demonstrated with various lighting configurations and various error bounds. In this chapter the method of sampling and reconstruction in [SKS02] is referred as the *ICP* method, though the ICP is only an algorithm of finding representative points.

Fig. 4.2 compares our sampling and reconstruction methods to that of ICP method. The demonstrated armadillo model has 10 002 vertices and 20 000 faces. Our methods capture local lighting variation details, while the ICP method misses the local color variations on the hands and face of the armadillo model. The lighting configuration is a square matrix of colored blocks, shown in Fig. 4.1. Fig. 4.3 shows the sample distribution generated by our methods and the ICP algorithm. Larger dots denote samples with larger valid domains. Darker areas will have less sample distribution. The samples generated by the ICP algorithm have no valid domains; so all dots are of equal size. See Table 4.1 for the comparison of errors and computational costs. The

	Groud truth	ICP		Our method	
				$\epsilon = 0.01$	$\epsilon = 0.0078$
Number of samples	10 002	128	256	133	250
Sampling cost (sec)	39.682	0.560	1.042	0.586	1.018
Recon. cost (sec)	0.167	0.500	0.775	0.067	0.073
RMSE	0	0.136	0.134	0.096	0.062

Table 4.1: Error and timing comparisons of the armadillo model in Fig. 4.2.

RMSE is the root mean square error that compares the color of ground truth with our methods and the ICP method, respectively. ϵ is the user-specified error bound of our methods to control the number of samples.

The sampling time in our methods consist of both the cost of hierarchy traversal and the time spent on sampling. The sampling time of the ICP method is the sampling time only, since the sample location is determined by the off-line ICP process. Though we have to traverse the hierarchy to place the samples, our overhead on hierarchy traversal is quite small. This is because most computation time is spent on reading back color buffers to main memory and on computing spherical harmonic coefficients. The sample number of the ground truth is exactly the vertex number of the armadillo model. The cost of sampling is nearly linear to the number of sample points. Note that as the number of samples increases, our interpolation method performs faster than the ICP method.



Figure 4.1: Lighting configuration in Fig. 4.2.



(a) Ground truth.

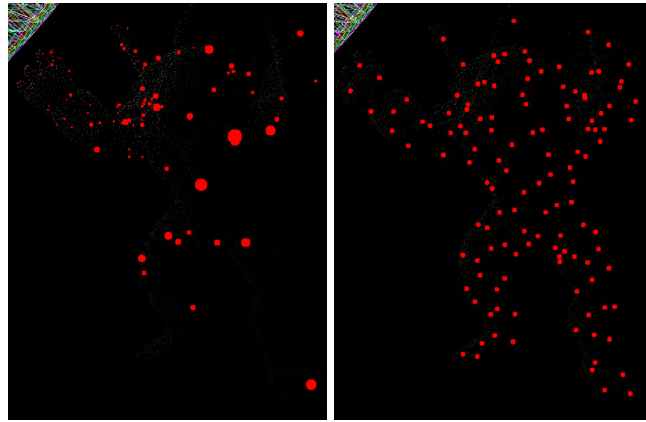


(b) Our methods with 133 samples. (c) Uniform sampling 128 samples with ICP method.



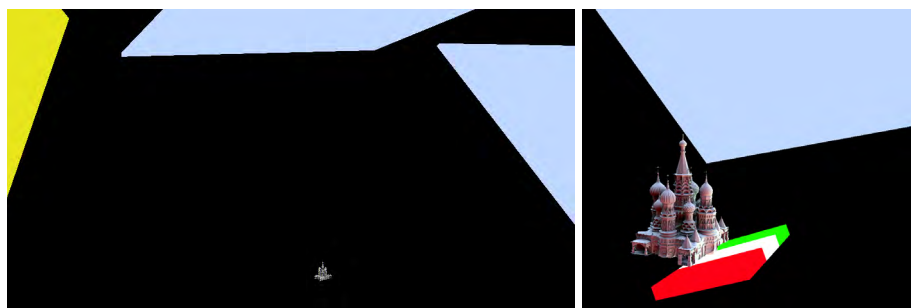
(d) Our methods with 250 samples. (e) Uniform sampling 256 samples with ICP method.

Figure 4.2: Comparisons of the armadillo model for our methods, the previous work with ICP, and the ground truth.



(a) Sample Distribution of our methods with 135 samples. (b) Sample Distribution of the ICP method with 128 samples.

Figure 4.3: Sampling distribution of our methods and the ICP algorithm.



(a) Lighting configuration: *far*. (b) Lighting configuration: *near*.

Figure 4.4: Different lighting configurations for the church model.

Fig. 4.5 shows the result of a more complex church model with 50 062 vertices and 99 486 faces and the lighting configuration as shown in Fig. 4.4(b). The lighting variations on the walls near the camera are missing for the ICP method. The error and timing statistics are shown in Table 4.2. Note that the numbers of samples computed by the ICP method are limited to the power of 2. Still, our methods effectively capture the variation of lighting. Also, our reconstruction method outperformed the original reconstruction method with a speed up ratio around ten.

Our methods adjust the number and distribution of samples under different lighting configurations. Fig. 4.4 demonstrates the same church model illuminated by two different lighting configurations—*near* and *far*. *Far* consists of three large planar emitters; and *near* consists of two large planar emitters and three nearby cubic emitters in red, white, and green. Fig. 4.6 shows that both lighting configurations rendered with our methods, with both the error bounds set to 0.001. Under different configurations, the visual appearances are both similar to the ground truth, but the number of samples are different. Table 4.3 shows the timings and sample numbers of both configurations. The church model contains 50 062 vertices. “GT time” is the sampling time of ground truth. “Sampling time” is the sampling time of our methods. Our methods place only one sample under the *far* configuration, and place 3899 samples under the *near* configuration.

Fig. 4.7 shows the ground truth of the Buddha model with 144 628 vertices and 293 232 faces, compared with the results of our methods under different user-specified error bounds. The number of samples and the RMSEs are shown in Table 4.4. As the specified error bound increases, the RMSE increases accordingly.

	Ground truth	Our method		ICP	
		$\epsilon = 0.007$	$\epsilon = 0.004$		
Number of samples	50 062	159	495	256	512
Sampling time (sec)	164.402	0.551	1.721	0.973	1.953
Recon. time (sec)	0.167	0.552	0.872	6.951	14.178
RMSE	0	0.037	0.016	0.129	0.115

Table 4.2: Error and timing comparison of the church model in Fig. 4.5.

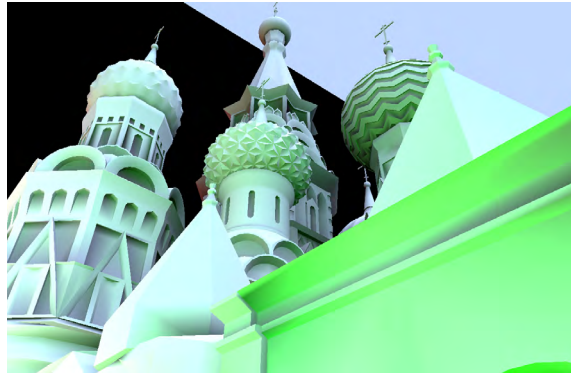
	<i>far</i>	<i>near</i>
Sampling time for ground truth images (sec)	161.403	164.406
Error bounds	0.001	0.001
Number of samples	1	3 899
Sampling time (sec)	0.042	13.544
Reconstruction time (sec)	0.119	3.922

Table 4.3: Sample numbers and costs under different lighting configurations with our method.

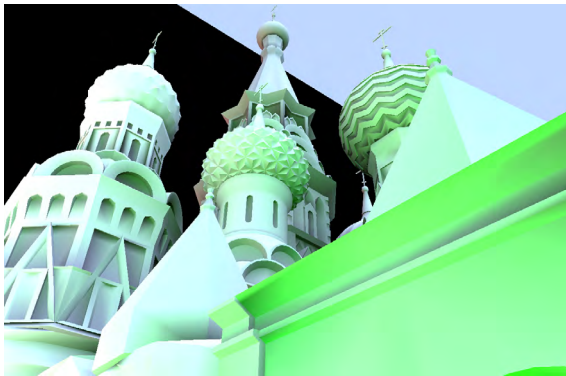


	Figure 4.7(b)	Figure 4.7(c)	Figure 4.7(d)
Error bounds	0.010	0.020	0.040
Number of samples	4 096	1 426	193
Sampling time (sec)	15.747	5.561	0.736
Recon. cost (sec)	6.597	2.102	0.933
RMSE	0.013	0.029	0.039

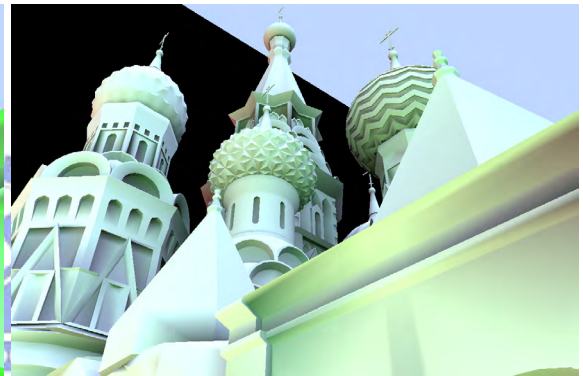
Table 4.4: Errors and timing statistics for Fig. 4.7.



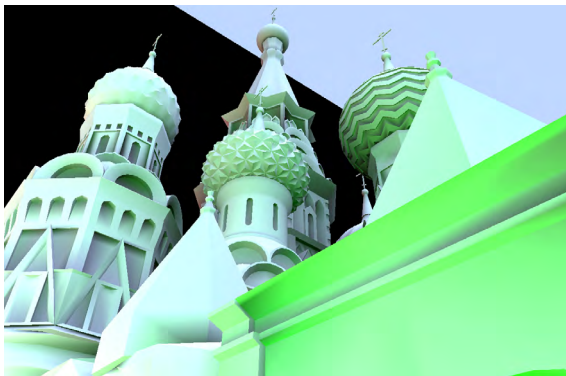
(a) Ground truth.



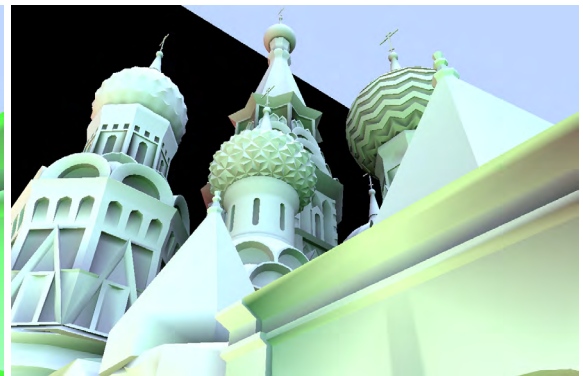
(b) Our methods with 154 samples.



(c) The ICP method with 256 samples.



(d) Our methods with 495 samples.



(e) The ICP method with 512 samples.

Figure 4.5: Comparisons of the church model for our methods with $\epsilon = 0.007$, $\epsilon = 0.004$, the ICP method with 256 samples, 512 samples, and the ground truth.

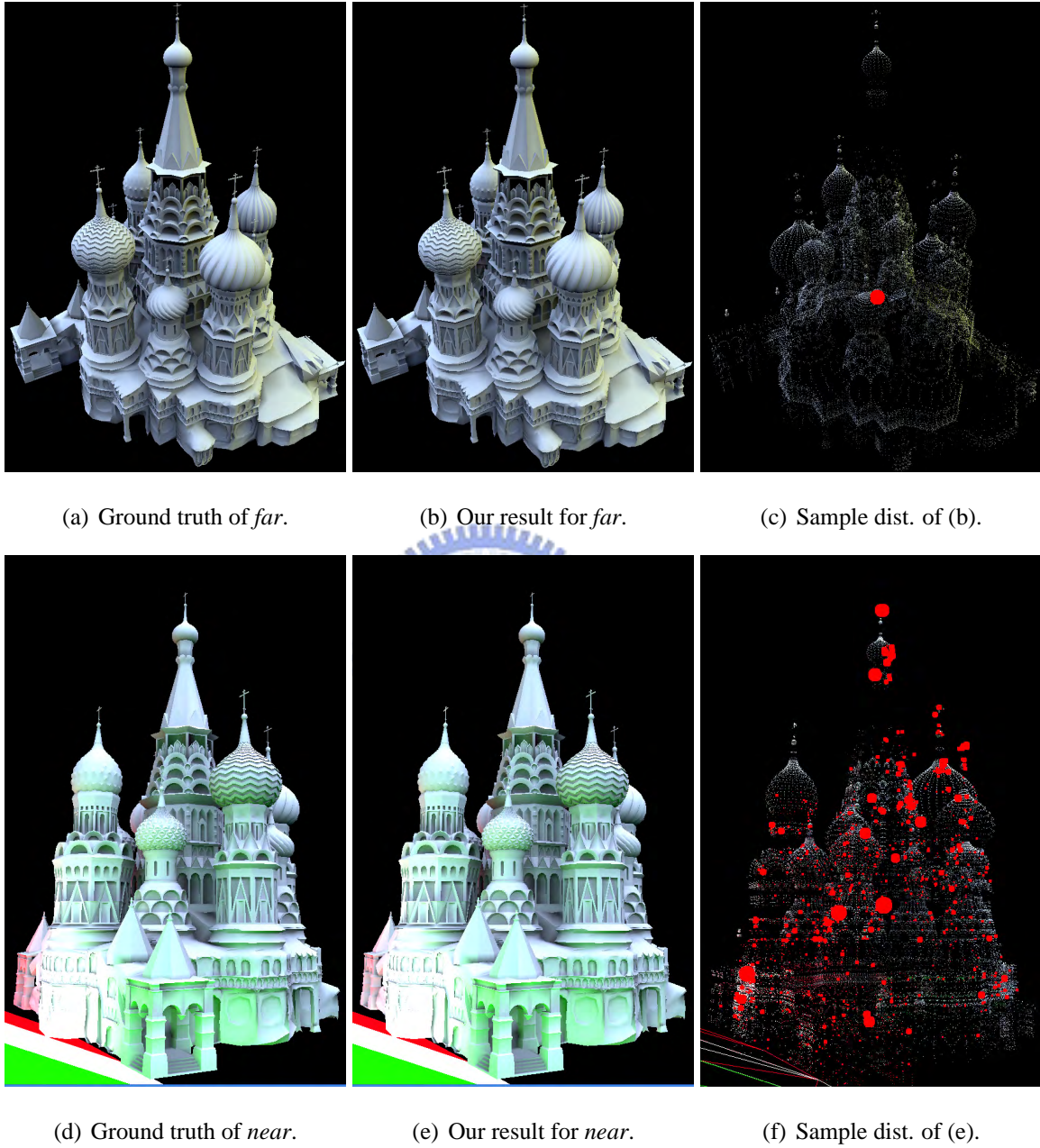
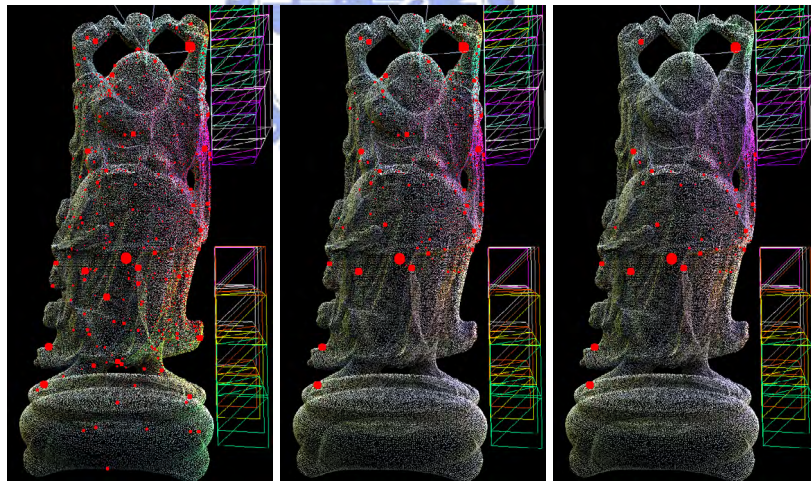


Figure 4.6: Our methods detect the lighting configurations, and the sample numbers under the same error bound are adjusted accordingly. Both lighting configurations are rendered with $\epsilon = 0.001$.



(a) Ground truth.

(b) $\epsilon = 0.01$ (c) $\epsilon = 0.02$ (d) $\epsilon = 0.04$ 

(e) Sample dist. of (b)

(f) Sample dist. of (c)

(g) Sample dist. of (d)

Figure 4.7: Results of different error bounds for the Buddha model.

Conclusion and Future Work

5.1 Summary

We present a hierarchical sampling framework that provides a user-controllable mechanism on rendering quality and speed for precomputed radiance transfer lighting. We tackle the problem of non-distant light sources by using the sparse sampling. A bounding volume hierarchy of the object is created as an off-line process. During run time, we traverse the BVH and evaluate the estimated error for sampling the incident radiance function at each node of the bounding volume hierarchy; If the estimated error is smaller than the user-specified error bound, a sample is assigned to the node; otherwise, children of the node are traversed recursively and more samples will be assigned for the node. The positions of samples are decided by averaging the position of the vertices inside the node. Each sample is also associated with a valid domain, by which the locality of lighting is preserved. Different from the previous works, our method can handle dynamic non-distant lighting and can be scalable to even mid-range or near-range emitters. Our interpolation scheme is also faster than that of the previous works by employing the valid domain and applying the linear octree for speedup.

5.2 Future Work

5.2.1 GPU Acceleration

The performance bottlenecks in our sampling implementation are reading back buffers containing the cubemaps and converting them into spherical harmonic coefficients. For reconstruction, the performance bottleneck is the computation of sample weights. If the sampling and reconstruction of our framework can be mapped onto the GPU, redundant bus transfers will be avoided and in consequence, a dramatic performance boost is possible.

5.2.2 Higher-order Interpolator

Higher-order interpolation of samples is also an interesting issue for further investigation. It is desirable to explore the possibilities of combining our sampling framework with a higher-order interpolator such as the work by Annen et al. [AKDS04]. With the high-order interpolation, the number of samples can be declined and the sample distribution may vary, and in consequence the oracle function may have to be adjusted accordingly.

5.2.3 Oracle Improvement

Deriving tighter sampling error bounds is also a challenging task. One feasible direction is to take the transfer function into account. Our oracle considers only the incident radiance function and some relevant information. However, since the color of vertices is decided by both the transfer function and the incident radiance function, the oracle should also predict possible variations in the transfer function. An extreme example is a black object that reflects no radiance: we should place no samples because the resulting color is always black, no matter how strong the incident radiance function is. Similar ideas can be found in the hierarchical refinement of clustering radiosity [SAG94, WHG99]. Different from the form factor computation, which is computed on the fly in the hierarchical radiosity methods, the spherical harmonic coefficients representing the transfer function of each vertex is precomputed in our framework. We may precompute some attributes of the coefficients at each vertex in every node of the bounding volume hierarchy. These attributes can be then used in the oracle.

Obtaining tighter error bounds is very important for rendering glossy surfaces, which is allowed in our framework. Rendering glossy surfaces in general requires tighter error bound

derived by an oracle that is able to predict the error on surfaces with glossy transfer functions. The oracle should be more sensitive to the variation of incident radiance and the viewing parameters. By this way the variations in vertex colors can be captured more accurately.

Though our framework works reasonably well for diffuse emitters, it cannot capture the variations of the incident radiance comes from specular emitters. Our oracle considers only the variations in the incident radiance that is approximated by the harmonic mean of distances to emitters, which may fail to deal with the specular emitters since the incident radiance on the receiver due to the specular emitters could still change rapidly even if the emitters are far away. How to deal with specular emitters is a challenging problem for most sparse sampling frameworks. The irradiance cache [WRC88] and the irradiance volume [GSHG98] both assume diffuse emitters. Fortunately, specular emitters does not abound in most real-time rendering applications. To handle the variation of the incident radiance due to specular emitters effectively, some kinds of light ray tracing is possible. However, most light ray tracing techniques such as [Jen01] can not be suited for, to some extent, the real-time rendering framework. Plausible sampling of incident radiance accounting for glossy surfaces as well as specular emitters are worthy to further research.

5.2.4 Real-time Rendering Applications

To fully integrate our method into real-time rendering applications, limitations on the spherical harmonic lighting should be removed. Our method still suffers from the static nature of precomputed transfer coefficients. The objects being shaded must be moved or rotated rigidly. No deformations are allowed. One possible solution is to parameterize the precomputed transfer functions in the same way as the deformation does; see [JF03]. To this end, our sampling hierarchy must be dynamically adjusted to account for the dynamic deformation. Moreover, most real-time rendering applications such as games will feature indoor environments. Under indoor environments, however, emitters such as lamps are always inside the scene's convex hull, thus violating the assumptions of precomputing radiance transfer. It would be valuable to extend spherical harmonic lighting to real-time rendering applications, especially for dynamic or indoor environments.

Bibliography

- [AKDS04] Thomas Annen, Jan Kautz, Frédo Durand, and Hans-Peter Seidel. Spherical harmonic gradients for mid-range illumination. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 331–336, June 2004.
- [Arv94] James Arvo. The irradiance jacobian for partially occluded polyhedral sources. In *Proceedings of ACM SIGGRAPH 1994*, pages 343–350, New York, NY, USA, 1994. ACM Press.
- [CBCG02] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. Light field mapping: Efficient representation and hardware rendering of surface light fields. *ACM Transactions on Graphics*, 21(3):447–456, July 2002.
- [CMS87] Brian Cabral, Nelson Max, and Rebecca Springmeyer. Bidirectional reflection functions from surface bump maps. In *Proceedings of ACM SIGGRAPH 1987*, volume 21, pages 273–281, July 1987.
- [CPC84] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of ACM SIGGRAPH 1984*, pages 137–145, New York, NY, USA, 1984. ACM Press.
- [DBB03] Philip Dutré, Philippe Bekaert, and Kavita Bala. *Advanced Global Illumination*. A K Peters Ltd., 2003.
- [DKS⁺03] Katja Daubert, Jan Kautz, Hans-Peter Seidel, Wolfgang Heidrich, and Jean-Michel Dischler. Efficient light transport using precomputed visibility. *IEEE Computer Graphics and Applications*, 23(3):28–37, 2003.
- [Eri05] Christer Ericson. *Real Time Collision Detection*. Morgan Kaufmann, 2005.

- [Gre03] Robin Green. Spherical harmonic lighting, the gritty details. Technical report, SCEA, January 2003.
- [GSHG98] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics & Applications*, 18(2):32–43, 1998.
- [HV04] Xuejun Hao and Amitabh Varshney. Real-time rendering of translucent meshes. *ACM Transactions on Graphics*, 23(2):120–142, April 2004.
- [Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A K Peters Ltd., 2001.
- [JF03] Doug L. James and Kayvon Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics*, 22(3):879–887, July 2003.
- [Kaj86] James T. Kajiya. The rendering equation. In *Proceedings of ACM SIGGRAPH 1986*, volume 20, pages 143–150, August 1986.
- [KSS02] Jan Kautz, Peter-Pike Sloan, and John Snyder. Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *13th Eurographics Workshop on Rendering*, 2002.
- [LBG80] Yoseph Linde, Andrés Buzo, and Robert M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communication*, pages 84–95, 1980.
- [LSS04] Xinguo Liu, Peter-Pike Sloan, Heung-Yeung Shum, and John Snyder. All-frequency precomputed radiance transfer for glossy objects. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 337–344, June 2004.
- [Max88] Nelson L. Max. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer*, 4(2):109–117, July 1988.
- [MGW01] Tom Malzbender, Dan Gelb, and Hans Wolters. Polynomial texture maps. In *Proceedings of ACM SIGGRAPH 2001*, pages 519–528, August 2001.
- [NRH04] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. Triple product wavelet integrals for all-frequency relighting. *ACM Transactions on Graphics*, 23(3):477–487, August 2004.

- [Ope99] OpenGL Architectural Review Board. *OpenGL Extension Registry*, 1999.
- [PH04] Mat Pharr and Greg Humphreys. *Physically Based Rendering*. Morgan Kaufmann, 2004.
- [RH01] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of ACM SIGGRAPH 2001*, pages 497–500, August 2001.
- [RH04] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for reflection. *ACM Transactions on Graphics*, 23(4):1004–1042, October 2004.
- [SAG94] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In *Proceedings of ACM SIGGRAPH 1994*, pages 435–442, July 1994.
- [SHHS03] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics*, 22(3):382–391, July 2003.
- [SKS02] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21(3):527–536, July 2002.
- [SP94] Francois X. Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, 1994.
- [Wat80] G Alistair Watson. *Approximation Theory*. John Wiley & Sons Ltd, 1980.
- [WH92] Gregory J. Ward and Paul S. Heckbert. Irradiance gradients. In *Proceedings of Eurographics Workshop on Rendering*, 1992.
- [WHG99] Andrew Willmott, Paul S. Heckbert, and Michael Garland. Face cluster radiosity. In *Eurographics Rendering Workshop 1999*, June 1999.
- [WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of ACM SIGGRAPH 1988*, pages 85–92, New York, NY, USA, 1988. ACM Press.