UNIVERSITY OF CALIFORNIA SAN DIEGO

Appearance Preserving Prefiltering for Rendering Complex Scenes

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Lifan Wu

Committee in charge:

Professor Ravi Ramamoorthi, Chair
Professor Manmohan Chandraker
Professor Hao Su
Professor Zhuowen Tu
Professor Shuang Zhao

2020

The Dissertation of Lifan Wu is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2020

DEDICATION

To My Family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Ravi Ramamoorthi for his patient guidance, insightful advice, and continuous support. Five years ago, he convinced me to come to UC San Diego as one of the first students in a new research group. During these five years, he has taught me to think like a scientist and do things with a high standard. It has been a great privilege to work with Ravi.

I am sincerely grateful to Shuang Zhao for his support in every aspect. He has always been available for discussions about technical questions. I would also like to thank my other committee members, Manmohan Chandraker, Hao Su, and Zhuowen Tu, for the inspiration and constructive advice they have provided through these years.

I want to thank all my collaborators: Ravi Ramamoorthi, Shuang Zhao, Lingqi Yan, Guangyan Cai, Frédo Durand, Ioannis Gkioulekas, Alexandr Kuznetsov, Cheng Zhang, Changxi Zheng, and Yang Zhou. I feel extremely fortunate to have the chance to work with such an excellent group of people. Special thanks to Lingqi Yan for the numerous hours we spent discussing exciting ideas.

I would like to thank my lab mates at UC San Diego, Sai Bi, Nima Khademi Kalantari, Alexandr Kuznetsov, Zhengqin Li, Kai-En Lin, Ishit Mehta, Zak Murez, Mohammad Shafiei, Tiancheng Sun, Weilun Sun, Jingwen Wang, Zexiang Xu, Jiyang Yu, and Shilin Zhu, for the countless conversations and the stimulating environment they created. I also want to thank my friends who have provided invaluable help over the years: Julaiti Alafate, Xiaojian Evans Chen, Zhao Fu, Yu Guo, Jian Jiang, Yanqin Jin, Wang-Cheng Kang, Suqi Liu, Jianmo Ni, Yao Qin, Saining Xie, Mengting Wan, Yizhen Wang, Songbai Yan, Zhen Zhai, Chicheng Zhang, Yufei Zhao and many others. In particular, I want to thank Sai Bi, Yao Qin, Mengting Wan, and Zexiang Xu for their support in life and work.

I would like to express my gratitude to my internship mentors: Mark Duchaineau at Google, Marios Papas and Jan Novák at Disney Research Zürich (Jan is at NVIDIA now), Chris Wyman, Marco Salvi and Aaron Lefohn at NVIDIA. Every internship was a wonderful

experience. I want to thank my undergraduate research advisors, Shi-Min Hu, Tao Ju, and Kun Xu, for introducing me to computer graphics and teaching me how to conduct research. Furthermore, I would like to thank Yiqing Zeng, my Olympiad in Informatics (OI) coach at high school, for showing me the beauty of computer programming.

Finally, I thank my parents Qiushan Wu and Wenzhen Huang for their unconditional love and unfaltering support. They give me the courage and wisdom to explore and pursue things I am interested in. This dissertation is dedicated to my family.

This work was supported in part by NSF grants 1451828, 1451830, 1703957, 1813553, and 1900927, the AWS Cloud Credits for Research, an NVIDIA Graduate Fellowship, the Ronald L. Graham Chair, and the UC San Diego Center for Visual Computing.

Chapter 3 is based on the material as it appears in ACM Transactions on Graphics, 2016 ("Downsampling Scattering Parameters for Rendering Anisotropic Media", Shuang Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 4 is based on the material as it appears in ACM Transactions on Graphics, 2019 ("Accurate Appearance Preserving Prefiltering for Rendering Displacement-Mapped Surfaces", Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 5 is based on the material to appear in ACM Transactions on Graphics, 2020 ("Analytic Spherical Harmonic Gradients for Real-Time Rendering with Many Polygonal Area Lights", Lifan Wu, Guangyan Cai, Shuang Zhao, and Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

VITA

2011–2015	B.Eng. in Computer Science, Tsinghua University
2015–2018	M.S. in Computer Science, University of California San Diego
2015–2020	Ph.D. in Computer Science, University of California San Diego

ABSTRACT OF THE DISSERTATION

Appearance Preserving Prefiltering for Rendering Complex Scenes

by

Lifan Wu

Doctor of Philosophy in Computer Science

University of California San Diego, 2020

Professor Ravi Ramamoorthi, Chair

Complex materials and lighting conditions can produce remarkably high-quality renderings with photorealism. However, efficiently rendering them remains challenging due to the high complexity of physically based light transport simulation. To improve the efficiency of rendering complex scenes, one common approach is using *prefiltering* techniques to precompute simplified material or lighting models. But usually these prefiltered models cannot *preserve* their original appearance accurately, leading to undesired rendering results.

In this dissertation, we present a series of novel *appearance preserving prefiltering* techniques. Our goal is to compute prefiltered material and lighting models that have smaller storage sizes and enable efficient rendering of complex scenes. Images rendered using the

prefiltered model should closely resemble the overall appearance given by the original model.

Our first work addresses the challenge of prefiltering volumetric scattering models. We present a joint optimization of multiple material scattering parameters, i.e., single-scattering albedos and phase functions, to accurately prefilter heterogeneous and anisotropic media. Our method leads to significantly better accuracy compared to traditional linear downsampling and offers several orders of magnitude storage reduction.

Our second work focuses on prefiltering the reflectance of a displacement-mapped surface. We represent the prefiltered surface reflectance model as a spatially varying bidirectional reflectance distribution function (SVBRDF) at a reduced resolution. To express our appearance preserving SVBRDF efficiently, we decompose it into a spatially varying normal distribution function (SVNDF) and a novel scaling function that accurately captures micro-scale changes of shadowing, masking, and interreflection effects. The scaling function can further be computed by an efficient factorization. Our prefiltering method generalizes well to different types of surfaces and enables anti-aliased level-of-detail rendering.

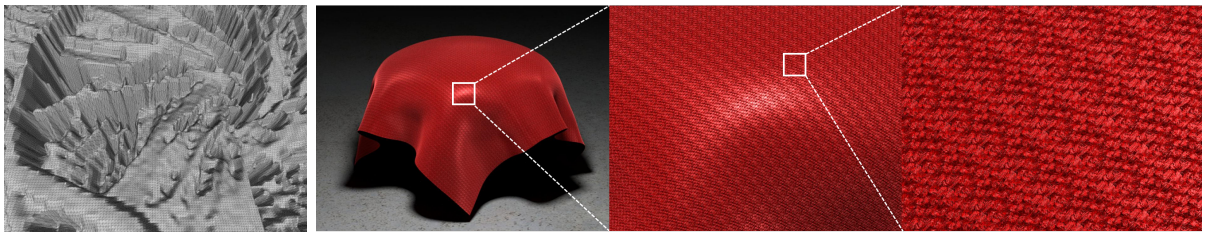
Finally, our last work turns to the challenge caused by complex lighting conditions. Given a scene with many polygonal area lights, we prefilter all the lights into a sparse 3D grid of spherical harmonic (SH) lighting coefficients and gradients, allowing interpolating SH coefficients accurately at any intermediate point. This enables scaling real-time precomputed radiance transfer (PRT) to hundreds of area lights. Our method builds on a novel analytic formula for SH gradients, which may benefit many other applications beyond rendering.

Chapter 1

Introduction

Photorealistic rendering is a challenging problem in computer graphics. Given a scene that contains 3D objects, scattering materials, light sources and camera configurations, our goal is to create an image that is indistinguishable from a photo taken with the same scene description, which is known as *photorealism*. To achieve photorealism, it usually requires physically based light transport simulation. The simulation includes light propagation and interaction events such as emitting from sources, traveling through media, interacting with scene objects, and finally reaching the camera. Accurately simulating light transport is computationally expensive, especially with complex scene geometries, materials, illumination conditions and camera models, which can produce various appealing visual effects. In this dissertation, we focus on efficient rendering techniques for complex materials and illuminations.

To accurately reproduce the appearance of complex materials such as cloth fabrics, researchers have developed micro-appearance models that explicitly describe the geometric structures at the microscopic scale (i.e., micro-geometry). These models can produce high-quality renderings with rich appearance details, especially when camera is at a close view (Figure 1.1). However, it is challenging to render these micro-appearance models efficiently. First, they are highly data intensive, because they are usually represented by high-resolution 2D or 3D textures. Second, a full light transport simulation on these models is very complicated due to the multiple scattering of light within the micro-geometries. Therefore, it requires significant



(a) Micro-geometry (b) Twill fabric renderings at multiple scales

Figure 1.1. Micro-appearance model of twill fabric. (a) We use a displacement map to represent the micro-geometry of the twill fabric, and visualize it as a mesh. (b) We render the twill fabric at multiple scales. The micro-appearance model is capable of reproducing characteristic visual effects such as glossy highlights at the macroscopic level and geometric details at the microscopic level.

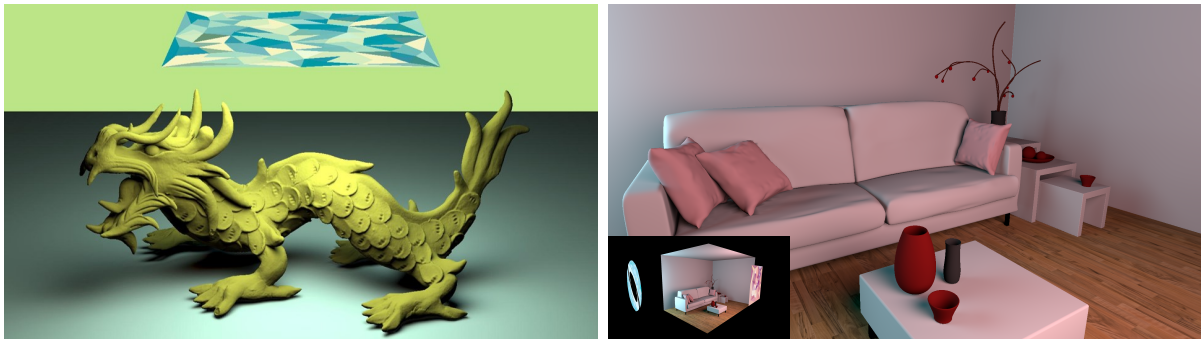


Figure 1.2. Renderings of scenes with multiple area lights.

storage and computation time to render high-quality images with micro-appearance models.

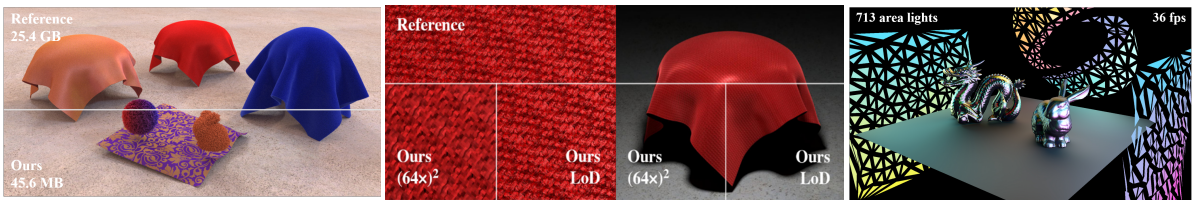
Another challenge in rendering comes from complex illumination conditions. It is common to have multiple light sources in modern computer graphics applications, as they can provide more natural and realistic lighting effects (Figure 1.2). But rendering scenes with many lights can be expensive, because the cost is usually linear in the number of light sources. The problem becomes even harder for real-time applications, in which the scene objects and light sources can change dynamically.

There has been much research in the past decades addressing the difficulty of rendering scenes with complex materials and illuminations. One commonly used technique is *prefiltering*, which precomputes simpler material or lighting models that enable more efficient rendering. However, using prefiltered rendering models usually causes accuracy degradation due to approxi-

mations made in the prefiltering processes. Thus, our goal is to apply prefiltering techniques to complex material and lighting models while preserving the appearance accuracy. That is, images rendered using the prefiltered models should look almost the same as those rendered using the original models.

In this dissertation, we first address the prefiltering of complex materials. For complex surface reflectance or volumetric scattering models, a prefiltering process usually involves downsampling the original high-resolution texture maps (e.g., 2D displacement maps or 3D volumetric grids) to lower-resolution ones. Prefiltering these models while accurately preserving their overall appearance remains challenging [7, 29]. This is because downsampling without careful consideration usually weakens the intrinsic self-shadowing structures of the micro-geometries, resulting in brightened appearance. A comprehensive analysis of prefiltering is very difficult, since the light interactions within the micro-geometries is highly nonlinear when we take global illumination (i.e., interreflection or multiple scattering) into account. There have been many prior attempts to solve this appearance prefiltering problem. But they are either neglecting global illumination effects [17, 25], or limited to micro-geometries that follow certain statistical distributions [29, 31]. In Chapters 3 and 4, we introduce novel appearance preserving prefiltering techniques for general micro-geometry configurations under global illumination. Our new prefiltering techniques not only lead to significant data storage reduction, but also enable accurate anti-aliased rendering at multiple scales.

Next, we present a prefiltering method for multiple area light sources. We use spherical harmonic (SH) lighting representations in a precomputed radiance transfer (PRT) framework [89]. Original PRT supports real-time dynamic low-frequency environmental (far-field) lighting, assuming the light sources are infinitely far away. Recent work [5, 102] has derived analytic SH lighting coefficients for a uniform polygonal area light, enabling close-up (near-field) area lighting in PRT. However, these methods require evaluating SH coefficients at each shading point for each area light. The running time is linear in the number of area lights, so scaling these methods to multiple area lights is prohibitively expensive. In Chapter 5, we demonstrate a novel



(a) Chapter 3 (b) Chapter 4 (c) Chapter 5

Figure 1.3. Overview of our contributions in this dissertation.

technique that prefilters multiple polygonal area lights into a sparse 3D grid of SH coefficients and gradients, enabling rendering scenes with hundreds of area lights in real-time. Our method builds on a novel theoretical result: analytic formulae for SH gradients.

To summarize, we develop a series of appearance preserving prefiltering techniques for efficient rendering of complex materials and lighting conditions. These techniques were presented at multiple ACM SIGGRAPH conferences [108, 109, 121]. Our specific contributions include:

Prefiltering volumetric scattering models. Volumetric micro-appearance models have provided remarkably high-quality renderings, but are highly data intensive and usually require tens of gigabytes in storage. When an object is viewed from a distance, the highest level of detail offered by these models is usually unnecessary, but traditional linear downsampling weakens the object's intrinsic shadowing structures and can yield poor accuracy. In Chapter 3, we introduce a joint optimization of single-scattering albedos and phase functions to accurately prefilter heterogeneous and anisotropic media represented by volumetric micro-appearance models (Figure 1.3(a)). Our method is built upon scaled phase functions, a new representation combining abledos and (standard) phase functions. We also show that modularity can be exploited to greatly reduce the amortized optimization overhead by allowing multiple synthesized models to share one set of downsampled parameters. Our optimized parameters generalize well to novel lighting and viewing configurations, and the resulting data sets offer several orders of magnitude storage savings.

Prefiltering displacement-mapped surfaces. Prefiltering the reflectance of a surface de-

scribed by a displacement map while preserving its overall appearance is challenging, as smoothing a displacement map causes complex changes of illumination effects such as shadowing-masking and interreflection. In Chapter 4, we introduce a new method that prefilters displacement maps and bidirectional reflectance distribution functions (BRDF) jointly and constructs spatially varying BRDFs (SVBRDF) at reduced resolutions. These SVBRDFs preserve the appearance of the input models by capturing both shadowing-masking and interreflection effects. To express our appearance-preserving SVBRDFs efficiently, we leverage a new representation that involves spatially varying normal distribution functions (NDF) and a novel scaling function that accurately captures micro-scale changes of shadowing, masking, and interreflection effects. Further, we show that the 6D scaling function can be factorized into a 2D function of surface location and a 4D function of direction. By exploiting the smoothness of these functions, we develop a simple and efficient factorization method that does not require computing the full scaling function. The resulting functions can be represented at low resolutions (e.g., 4^2 for the spatial function and 15^4 for the angular function), leading to minimal additional storage. Our method generalizes well to different types of geometries beyond Gaussian surfaces. Models prefiltered using our approach at different scales can be combined to form mipmaps, allowing accurate and anti-aliased level-of-detail (LoD) rendering (Figure 1.3(b)).

Prefiltering multiple polygonal area lights. In Chapter 5, we introduce a novel algorithm that encodes light contributions from multiple uniform polygonal area lights in spherical harmonic (SH) coefficients and gradients stored on a sparse grid. This enables scaling PRT to hundreds of area lights with real-time frame rates (Figure 1.3(c)). To achieve this, we develop a novel analytic formula for the spatial gradients of the SH coefficients for uniform polygonal area lights. The result is a significant generalization, involving the Reynolds transport theorem to reduce the problem to a boundary integral for which we derive a new analytic formula, showing how to reduce a key term to an earlier recurrence for SH coefficients. The implementation requires only minor additions to existing code for SH coefficients. The theoretical results also

hold implications for recent efforts on differentiable rendering. We show that SH gradients enable very sparse spatial sampling, followed by accurate Hermite interpolation. In addition to efficient many-light rendering in PRT, the SH gradient formula is a new mathematical result that potentially enables many other graphics applications.

The dissertation is organized as follows. We first introduce the basic background on light transport and spherical harmonics in Chapter 2. From Chapters 3 to 5, we present technical details of our appearance preserving prefiltering methods for rendering volumetric scattering models, displacement-mapped surfaces, and multiple polygonal area lights, respectively. Finally, we present our conclusion and discuss future research directions in Chapter 6.

Chapter 2

Background

In this chapter, we review the basic background knowledge that is frequently used throughout this dissertation. We start with a brief review on physically based light transport (Section 2.1). Then, we introduce spherical harmonics and their key properties (Section 2.2), which are widely used in rendering applications.

2.1 Light Transport

To create photorealistic images, it is necessary to simulate light transport according to physical laws. In this dissertation, we use *geometric optics* models to formulate the behavior of light, neglecting more complicated wave and quantum phenomena. Geometric optics is usually sufficient for rendering applications. In addition, we assume that the environment we simulate is at the steady state. That is, light travels very fast and the radiance distribution in a scene reaches equilibrium after a very short time. Given these assumptions, we can formulate light radiance as a 5D function of position and direction, without considering the wavelength (spectral) and time dimensions.

A key concept of light transport is the interaction between light and scene objects. In the rest of this section, we first introduce basic theories of surface reflection (Section 2.1.1) and volume scattering (Section 2.1.2). Next, we briefly review several practical techniques to solve light transport simulation (Section 2.1.3).

2.1.1 Surface Reflection

When light hits a surface, it reflects back to the scene with attenuated energy. Kajiya's *rendering equation* [48] formulates surface reflection as

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\mathcal{H}^2} f_r(\boldsymbol{x}, \boldsymbol{\omega}' \to \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega}') \langle \boldsymbol{n}(\boldsymbol{x}), \boldsymbol{\omega}' \rangle \, \mathrm{d}\boldsymbol{\omega}'. \tag{2.1}$$

In this equation, the quantity $L(\boldsymbol{x}, \boldsymbol{\omega})$ describes the outgoing radiance along direction $\boldsymbol{\omega}$ at a specific point $\boldsymbol{x}$. On the right-hand side, it is the sum of the emitted radiance $L_e(\boldsymbol{x}, \boldsymbol{\omega})$ from the surface and an integral that expresses the reflected radiance. The domain of integration is the upper hemisphere $\mathcal{H}^2$ on the surface. Within the integrand, the bidirectional reflectance distribution function (BRDF) $f_r(\boldsymbol{x}, \boldsymbol{\omega}' \to \boldsymbol{\omega})$ describes the ratio between reflected radiance and irradiance at the point $\boldsymbol{x}$, $L_i(\boldsymbol{x}, \boldsymbol{\omega}')$ denotes the incoming radiance from direction $\boldsymbol{\omega}'$, $\boldsymbol{n}(\boldsymbol{x})$ is the normal direction, and $\langle \cdot, \cdot \rangle$ represents dot product.

Note that by convention, all the directions in the radiance quantities and BRDFs point away from $\boldsymbol{x}$. Specifically, the relation between the incoming radiance L_i and the outgoing radiance L can be written as

$$L_i(\boldsymbol{x}, \boldsymbol{\omega}) = \lim_{t \to 0^+} L(\boldsymbol{x} + t\boldsymbol{\omega}, -\boldsymbol{\omega}). \tag{2.2}$$

When a scene does not have participating media, i.e., objects are in a vacuum, the rendering equation can fully capture the light transport process. We are able to compute an image by solving this equation. In the next section, we will discuss volumetric light transport in scenes with participating media.

2.1.2 Volume Scattering

Modeling light propagation and interactions inside participating media can give us a large variety of visual effects that are impossible to be generated with only surface reflection, e.g.,

atmospheric effects of fog or translucent appearance of jade. Simulating volume scattering has become an important component in modern production rendering applications. In Chapter 3, we use participating media to model the appearance of fabrics and fur. We encourage the enthusiastic reader to consult the survey of Novák et al. [74] for a comprehensive review of volumetric light transport simulation.

We start with an introduction to the relevant optical parameters of participating media. From a statistical point of view, there are particles, which either absorb or scatter light, randomly distributed in a participating medium. We denote σ_a and σ_s as the absorption and scattering coefficient, respectively. They represent the probability densities of a photon being absorbed or scattered after traveling a unit distance. In addition, we define the extinction coefficient $\sigma_t := \sigma_a + \sigma_s$ and the single-scattering albedo $\alpha := \sigma_s/\sigma_t$. As an analogue of BRDFs in surface reflection, we use phase functions $f_p(\boldsymbol{\omega}' \to \boldsymbol{\omega})$ to describe the directional distribution of light scattered in a medium.

When light travels inside a participating medium, there are four types of light-medium interaction events: emission, absorption, out-scattering, and in-scattering. The *radiative transfer equation* (RTE) [9] describes how the radiance changes by these interaction events:

$$(\boldsymbol{\omega} \cdot \nabla) L(\boldsymbol{x}, \boldsymbol{\omega}) = -\underbrace{\sigma_a L(\boldsymbol{x}, \boldsymbol{\omega})}_{\text{absorption}} - \underbrace{\sigma_s L(\boldsymbol{x}, \boldsymbol{\omega})}_{\text{out-scattering}} + \underbrace{\sigma_s L_{\text{ins}}(\boldsymbol{x}, \boldsymbol{\omega})}_{\text{in-scattering}} + \underbrace{\sigma_a L_e(\boldsymbol{x}, \boldsymbol{\omega})}_{\text{emission}}, \tag{2.3}$$

where the in-scattered radiance is an integral over all directions on the unit sphere $\mathcal{S}^2$:

$$L_{\text{ins}}(\boldsymbol{x}, \boldsymbol{\omega}) = \int_{\mathcal{S}^2} f_p(\boldsymbol{x}, \boldsymbol{\omega}' \to \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}'. \tag{2.4}$$

With the rendering equation (2.1) at surface points as boundary conditions, we can rewrite the differential RTE as a pure integral equation by integrating both sides of the RTE (2.3). The

resulting *volume rendering equation* is

$$L(\boldsymbol{x},\boldsymbol{\omega}) = Q(\boldsymbol{x},\boldsymbol{\omega}) + \int_0^D T(\boldsymbol{x}',\boldsymbol{x})\sigma_s(\boldsymbol{x}')L_{\text{ins}}(\boldsymbol{x}',\boldsymbol{\omega})\,\mathrm{d}\tau, \tag{2.5}$$

where $Q(\boldsymbol{x},\boldsymbol{\omega})$ is the source term that accounts for self-emissions on the surface and inside the medium, D is distance from x to the nearest surface in direction $-\boldsymbol{\omega}$, point $\boldsymbol{x}' := \boldsymbol{x} - \tau\boldsymbol{\omega}$ is between $\boldsymbol{x}$ and the surface boundary, and $T(\boldsymbol{x}',\boldsymbol{x})$ is the transmittance between $\boldsymbol{x}'$ and $\boldsymbol{x}$:

$$T(\boldsymbol{x}',\boldsymbol{x}) = \exp\left(-\int_0^\tau \sigma_t(\boldsymbol{x}-\tau'\boldsymbol{\omega})\,\mathrm{d}\tau'\right). \tag{2.6}$$

Incorporating the rendering equation (2.1), we can express the source term $Q(\boldsymbol{x},\boldsymbol{\omega})$ as

$$Q(\boldsymbol{x},\boldsymbol{\omega}) = \int_0^D T(\boldsymbol{x}',\boldsymbol{x})\sigma_a(\boldsymbol{x}')L_e(\boldsymbol{x}',\boldsymbol{\omega})\,\mathrm{d}\tau + T(\boldsymbol{x}_0,\boldsymbol{x})L(\boldsymbol{x}_0,\boldsymbol{\omega}), \tag{2.7}$$

where $\boldsymbol{x}_0 := \boldsymbol{x} - D\boldsymbol{\omega}$ is the point on the nearest surface.

2.1.3 Path Integral Formulation

To solve the rendering equation (2.1), Veach presented the path integral formulation [99] by recursively expanding the original integral. Pauly et al. further extended this formulation to the volumetric rendering equation [80]. In the path integral formulation of (volumetric) light transport, our goal is to solve an integral

$$I = \int_\Omega f(\bar{x})\,\mathrm{d}\mu(\bar{x}) \tag{2.8}$$

that enumerates over the path space Ω containing all the possible transport paths in a scene. A transport path $\bar{x} = (\boldsymbol{x}_0,\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k)$ of length k is expressed by a sequence of vertices that are either on the surfaces or in the participating media. We denote $f(\bar{x})$ as the measurement contribution function of this path. The path throughput measure $\mathrm{d}\mu(\bar{x})$ is the product of measures at all the

vertices. In particular, the measurement contribution of a path $\bar{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ [74] can be expressed as

$$f(\bar{x}) = W(\boldsymbol{x}_0, \boldsymbol{x}_1) L_e(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}) G(\boldsymbol{x}_0, \boldsymbol{x}_1) T(\boldsymbol{x}_0, \boldsymbol{x}_1) \prod_{i=1}^{k-1} f_s(\boldsymbol{x}_{i-1}, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}) G(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) T(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}). \tag{2.9}$$

Assuming that $\boldsymbol{x}_0$ is on the sensor and $\boldsymbol{x}_k$ is on the light sources, the function $W(\boldsymbol{x}_0, \boldsymbol{x}_1)$ indicates the sensor response value. All the remaining terms are defined as follows, depending on the location of each path vertex:

$$\boldsymbol{\omega}_{\boldsymbol{x}\to\boldsymbol{y}} := \frac{\boldsymbol{y}-\boldsymbol{x}}{\|\boldsymbol{y}-\boldsymbol{x}\|}. \tag{2.10}$$

$$G(\boldsymbol{x}, \boldsymbol{y}) := \frac{D(\boldsymbol{x}, \boldsymbol{y}) D(\boldsymbol{y}, \boldsymbol{x})}{\|\boldsymbol{x}-\boldsymbol{y}\|^2}, \text{ where} \tag{2.11}$$

$$D(\boldsymbol{x}, \boldsymbol{y}) := \begin{cases} \|\boldsymbol{n}(\boldsymbol{x}) \cdot \boldsymbol{\omega}_{\boldsymbol{x}\to\boldsymbol{y}}\| & \text{if } \boldsymbol{x} \text{ is on a surface,} \\ 1 & \text{if } \boldsymbol{x} \text{ is in a medium.} \end{cases} \tag{2.12}$$

$$L_e(\boldsymbol{x}, \boldsymbol{y}) := \begin{cases} L_e(\boldsymbol{x}, \boldsymbol{\omega}_{\boldsymbol{x}\to\boldsymbol{y}}) & \text{if } \boldsymbol{x} \text{ is on a surface,} \\ \sigma_a(\boldsymbol{x}) L_e(\boldsymbol{x}, \boldsymbol{\omega}_{\boldsymbol{x}\to\boldsymbol{y}}) & \text{if } \boldsymbol{x} \text{ is in a medium.} \end{cases} \tag{2.13}$$

$$f_s(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) := \begin{cases} f_r(\boldsymbol{y}, \boldsymbol{\omega}_{\boldsymbol{y}\to\boldsymbol{x}}, \boldsymbol{\omega}_{\boldsymbol{y}\to\boldsymbol{z}}) & \text{if } \boldsymbol{x} \text{ is on a surface,} \\ \sigma_s(\boldsymbol{y}) f_p(\boldsymbol{y}, \boldsymbol{\omega}_{\boldsymbol{y}\to\boldsymbol{x}}, \boldsymbol{\omega}_{\boldsymbol{y}\to\boldsymbol{z}}) & \text{if } \boldsymbol{x} \text{ is in a medium.} \end{cases} \tag{2.14}$$

Because of the high complexity of the path integral, it is almost impossible to derive a general analytic solution. In practice, we utilize Monte Carlo integration to compute this integral numerically. An estimator of the path integral (2.8) can be written as

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{x}_i)}{p(\bar{x}_i)}. \tag{2.15}$$

Using this Monte Carlo method, a collection of transport paths $\bar{x}_i$ are sampled (i.e., constructed

by certain procedures such as path tracing) according to a probability density function $p(\cdot)$. As the number of path samples increases, the value $\langle I \rangle$ will converge to I eventually.

Monte Carlo integration techniques have been commonly used in modern physically based rendering algorithms due to their simplicity. To evaluate an integral in any dimension, the Monte Carlo estimator converges at a rate of $O(N^{-1/2})$, avoiding the curse of dimensionality. In addition, we can apply many variance reduction techniques such as importance sampling and control variates. Different estimators can be combined via multiple importance sampling (MIS) [99] to further reduce variance and accelerate convergence [19, 24, 56].

2.2 Spherical Harmonics

Spherical harmonics (SH) are a set of orthogonal functions defined on the unit sphere $\mathcal{S}^2$. They are analogous to 1D sinusoids in Fourier analysis. Since many rendering concepts involve spherical functions and integrals (e.g., BRDFs and the rendering equation (2.1)), spherical harmonics are widely used in rendering applications [83, 89]. In this section, we present a brief introduction to spherical harmonics and their key properties.

2.2.1 Definition

Let $\boldsymbol{\omega} = (x, y, z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$ be a unit direction on $\mathcal{S}^2$. The real SH basis functions for $l \geq 0, -l \leq m \leq l$ are [68]

$$Y_{lm}(\boldsymbol{\omega}) = \begin{cases} \sqrt{2} K_{lm} \sin(|m|\phi) P_l^{|m|}(\cos\theta), & m < 0, \\ K_{l0} P_l^0(\cos\theta), & m = 0, \\ \sqrt{2} K_{lm} \cos(m\phi) P_l^m(\cos\theta), & m > 0, \end{cases} \tag{2.16}$$

where P_l^m are the associated Legendre polynomials and K_{lm} is the normalization term, given by

$$K_{lm} = \sqrt{\frac{2l+1}{4\pi} \frac{(l-|m|)!}{(l+|m|)!}}. \tag{2.17}$$

In particular, zonal harmonics (ZH) are a subset of SH basis functions for $m = 0$:

$$Y_{l0}(\boldsymbol{\omega}) = K_l P_l(\cos\theta), \tag{2.18}$$

where the normalization $K_l = \sqrt{\frac{2l+1}{4\pi}}$ and P_l is the Legendre polynomial of degree l. Zonal harmonics are radially symmetric around the z-axis. To represent a ZH basis function that is centered around an arbitrary axis $\boldsymbol{\omega}_c$, we write the rotated ZH using dot products:

$$Y_{l0}(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_c) = K_l P_l(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_c). \tag{2.19}$$

SH basis functions are usually indexed by l and m. Sometimes it is more convenient to use a single-parameter indexing: $i = l(l+1)+m$. For notation simplicity, we will this indexing scheme in the next section.

2.2.2 Key Properties

SH projection and expansion. Given a real-valued function f defined on $\mathcal{S}^2$, we can compute the SH coefficients F_i by projecting f onto each individual SH basis function Y_i:

$$F_i = \int_{\mathcal{S}^2} f(\boldsymbol{\omega}) Y_i(\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega}. \tag{2.20}$$

The function f can be expanded as a linear combination of the SH basis functions, weighted by the SH coefficients:

$$f(\boldsymbol{\omega}) = \sum_{i=0}^{\infty} F_i Y_i(\boldsymbol{\omega}). \tag{2.21}$$

In practice, we typically set a number $l_{\max}$ to limit the number of SH coefficients, i.e., $0 \le l \le l_{\max}$. This leads to $N = (l_{\max}+1)^2$ coefficients. Since the high-frequency components

of f are truncated, we have a band-limited approximation $\tilde{f}$ of the original function:

$$f(\boldsymbol{\omega}) \approx \tilde{f}(\boldsymbol{\omega}) = \sum_{i=0}^{N-1} F_i Y_i(\boldsymbol{\omega}). \tag{2.22}$$

Double product integral. Given two spherical functions expanded with SH basis,

$$f(\boldsymbol{\omega}) = \sum_i F_i Y_i(\boldsymbol{\omega}),\ g(\boldsymbol{\omega}) = \sum_j G_j Y_j(\boldsymbol{\omega}), \tag{2.23}$$

we can compute the integral of their product as

$$\begin{aligned}
\int_{\mathcal{S}^2} f(\boldsymbol{\omega}) g(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} &= \int_{\mathcal{S}^2} \left(\sum_i F_i Y_i(\boldsymbol{\omega}) \right) \left(\sum_j G_j Y_j(\boldsymbol{\omega}) \right) \mathrm{d}\boldsymbol{\omega} \\
&= \sum_i \sum_j F_i G_j \underbrace{\int_{\mathcal{S}^2} Y_i(\boldsymbol{\omega}) Y_j(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega}}_{=: C_{ij}} \\
&= \sum_i \sum_j F_i G_j \delta_{ij} \\
&= \sum_i F_i G_i.
\end{aligned} \tag{2.24}$$

Here, C_{ij} are the coupling coefficients. Because the SH basis functions are orthonormal, we have $C_{ij} = \delta_{ij}$, where δ_{ij} is the Kronecker delta. Therefore, the double product integral is simply a dot product of their SH coefficient vectors. It is commonly used for evaluating diffuse reflections in precomputed radiance transfer (PRT) [89].

Triple product integral. In certain situations, we may need to compute a triple product integral, e.g., the rendering equation involving lighting, glossy BRDFs, and visibility [72, 89].

Given three SH expanded functions f, g, h as input, we can compute the triple product integral as

$$\begin{aligned}\int_{\mathcal{S}^2} f(\boldsymbol{\omega})g(\boldsymbol{\omega})h(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega} &= \int_{\mathcal{S}^2} \left(\sum_i F_i Y_i(\boldsymbol{\omega})\right)\left(\sum_j G_j Y_j(\boldsymbol{\omega})\right)\left(\sum_k H_k Y_k(\boldsymbol{\omega})\right)\mathrm{d}\boldsymbol{\omega} \\ &= \sum_i \sum_j \sum_k F_i G_j H_k \underbrace{\int_{\mathcal{S}^2} Y_i(\boldsymbol{\omega})Y_j(\boldsymbol{\omega})Y_k(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}}_{=:C_{ijk}} \\ &= \sum_i \sum_j \sum_k C_{ijk} F_i G_j H_k. \end{aligned} \tag{2.25}$$

The tripling coefficients C_{ijk} are known as the Clebsch-Gordan coefficients [38], which have analytic forms. Although the triple product integral has a more complicated solution than the double product integral, the coefficients C_{ijk} are sparse, and it leads to a computational complexity of $O(N^{5/2})$ [72]. This enables practical glossy reflections in PRT [72, 89].

Chapter 3

Downsampling Scattering Parameters for Rendering Anisotropic Media

We start with prefiltering the volumetric scattering parameters of participating media. Our goal is to compute the prefiltered scaled phase functions at reduced resolutions. We formulate prefiltering as an inverse rendering problem and present a practical optimization algorithm to solve it. Our method can produce prefiltered material models that accurately preserve the desired appearance and have significantly smaller data size than the original high-resolution models.

3.1 Introduction

Recently, detailed volumetric models [42, 118] have become increasingly more popular for handling the appearance of materials with complex 3D structures (e.g., fabric and fur). These *micro-appearance* models explicitly capture objects' microscopic level geometries and have brought the quality of computer rendered complex materials to the next level.

Unfortunately, efficient use of these high-resolution models remains challenging since they are usually highly data intensive, and loading these datasets into memory alone can take minutes. For many computer graphics applications, however, such extreme high-resolution is unnecessary: especially when the object is viewed from a distance. Greatly downsampled versions, which require significantly less storage, would suffice in many situations. In Figure 3.1, for instance, the reference scene involves 4.68×10^{12} effective voxels taking 25.4 GB of storage.

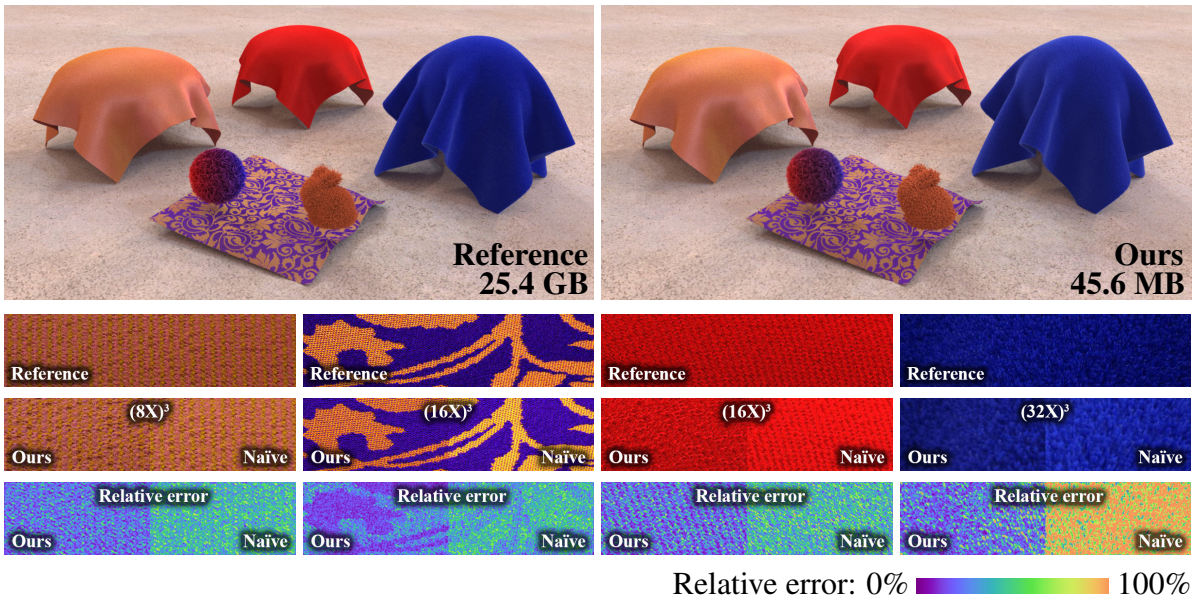


Figure 3.1. We present a new approach to compute scattering parameters at reduced resolutions. Many detailed appearance models involve high-resolution volumetric representations (top-left). Such level of detail leads to high storage but is usually unnecessary especially when the object is rendered at a distance. However, naïve downsampling often loses intrinsic shadowing structures and brightens resulting images (see the insets). Our method computes scaled phase functions, a combined representation of single-scattering albedo and phase function, and provides significantly better accuracy while reducing the data size by almost three orders of magnitude (top-right).

Our models with reduced resolutions (see the second row of insets for levels of downsampling), on the other hand, result in only 45.6 MB of data.

Although lower-resolution operations have been developed for plain textures and normal maps [25], they are largely lacking for light scattering parameters [9, 42]. Recently, Heitz et al. [29] proposed a new phase function formulation which offers easy (trilinear) prefiltering and is a valuable first-step toward this direction. However, linear downsampling is generally insufficient when handling scattering parameters including spatially varying densities and albedo values. This is because downsampling weakens intrinsic shadowing structures (Figure 3.2), causing significant brightening of resulting renderings (see the middle row of insets in Figure 3.1).

In this chapter, we introduce a novel method to improve the accuracy of downsampled

scattering parameters. In particular, our approach optimizes single-scattering albedos and phase functions jointly. Our contributions include:

- We introduce *scaled phase functions* combining albedos and phase functions (Section 3.4).
- We develop an optimization-based method to *downsample* scaled phase functions (Sections 3.5 and 3.6).
- We show how *modularity* can be exploited by reusing a single set of optimized parameters for multiple objects, significantly reducing the amortized optimization overhead (Section 3.7).

Our approach is model-dependent: unlike traditional prefiltering methods, our technique takes full input geometries into consideration. Our optimization involves a few training renderings which can take tens of CPU core hours but is comparable to rendering one high-quality image. Fortunately, this process only needs to be executed once: the resulting representations can be reused in different virtual configurations. Furthermore, by exploiting modularity, multiple objects can share one single optimization. We also release all our downsampled datasets so that the detailed models in Figures 3.1, 3.18, and 3.19 can be directly used by other researchers.

Our downsampled models offer several orders of magnitude storage saving and greatly reduce the I/O time, which can be a significant overhead for highly detailed scenes like Figure 3.1. In addition, our models are less prone to aliasing and usually require fewer sample paths to achieve similar rendering qualities (Figure 3.3). Lastly, these benefits are "effort-free": our technique allows the user to replace original parameters using significantly downsampled versions without modifying the core rendering algorithm.

3.2 Related Work

Radiative transfer. The scattering parameters considered in this chapter arise from radiative transfer [9] which originally assumes random media with disorganized micro-structures. Jakob et al. [42] later relaxed this assumption by modeling anisotropic media with structured micro-geometries. Heitz et al. [29] has recently proposed a new anisotropic phase function

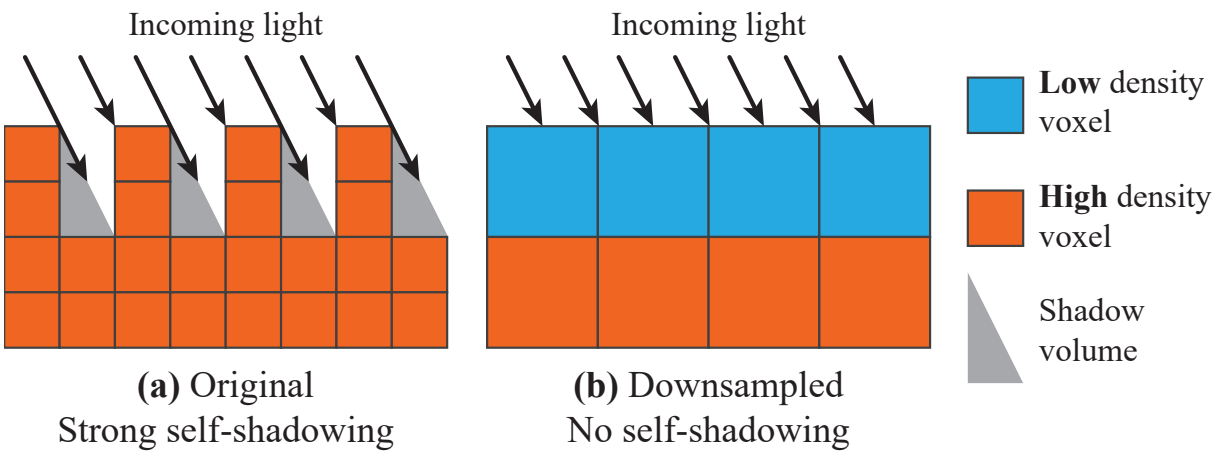


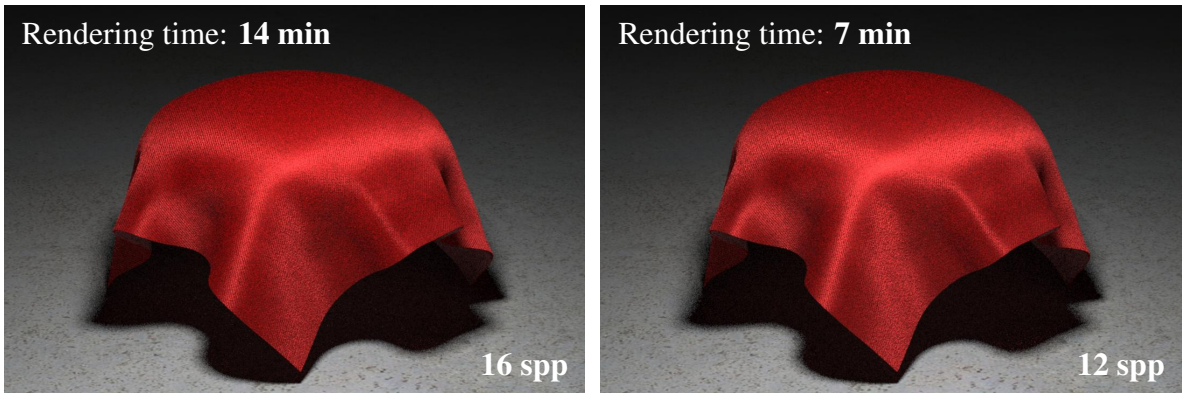
Figure 3.2. 2D demonstration of the weakening of self-occlusion caused by downsampling spatially varying densities.

offering fast evaluation and easy (linear) interpolation. We build our framework upon this state-of-the-art representation.

Micro-appearance models. Recently, a family of *micro-appearance modeling techniques* [118, 119] have been introduced. These methods represent material geometries at unprecedented detail using ultra high-resolution volumes. Leveraging anisotropic radiative transfer, they have provided extremely high-quality renderings. However, the use of large 3D volumes has yielded multi-gigabyte models. This chapter focuses on greatly reducing the resolution of these models while maintaining good accuracy (Figures 3.1, 3.18, 3.19).

Handling high-resolution data/appearance. Several techniques [51, 88] have been developed previously to interpolate voxelized data for efficient 3D visualization. Although these approaches could be adapted to handle some scattering parameters, they normally offer limited accuracy. In addition, a number of techniques have been introduced to efficiently handle surface-based reflectance profiles [7, 44, 95], micro-geometry [14, 43, 105]), or specialized volumes [70]. Unfortunately, these methods are not easily applicable to our problem.

Altering material properties. This chapter aims to compute a material's scattering parameters at reduced resolutions. Previously, a family of techniques focusing on altering material scattering parameters are similarity relations [120]. These methods, however, need to



(a) Original
Noise level: 2.35×10^{-4}

(b) $(16\times)^3$ Downsampled
Noise level: 2.47×10^{-4}

Figure 3.3. Equal-quality renderings using volumes at original (a) and $(16\times)^3$-downsampled (b) resolutions (computed with our method). The lower-resolution representation requires fewer samples per pixel to obtain a similar resulting noise level (measured using [64]).

be applied independently to each point and generally do not allow changing model resolutions. Our technique is completely orthogonal and complementary to them.

Inverse rendering. Inverse rendering techniques solve for material optical parameters given desired object appearance. Some of them are also based on optimizations [22, 26, 50]. However, these methods effectively solve a special case of our problem. That is, they assume homogeneity (i.e., spatially invariant parameters) and/or simple (single-lobe) phase functions. Our approach performs a joint optimization of both albedo and phase function, and supports heterogeneity.

3.3 Background

At the core of the radiative transfer framework [9] lies the *radiative transfer equation* (RTE)[1]

$$(\boldsymbol{\omega} \cdot \nabla) L(\boldsymbol{\omega}) = -\sigma_t L(\boldsymbol{\omega}) + \sigma_s \int_{\mathcal{S}^2} L(\boldsymbol{\omega}') f(\boldsymbol{\omega}' \to \boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega}', \tag{3.1}$$

[1] A source term $Q(\boldsymbol{\omega})$ from the RHS of Equation (3.1) is neglected as we focus on nonemissive media. In addition, spatial dependencies of L, σ_t, σ_s, and f are omitted for notational convenience.

Table 3.1. Definitions of commonly used symbols in Chapter 3.

Symbol	Meaning	Def.
σ_t	Extinction coefficient (density)	Section 3.3
σ_s	Scattering coefficient	Section 3.3
α	Single-scattering albedo	Section 3.3
f	Phase function	Section 3.3
$f^{\mathrm{SGGX}}(S)$	SGGX function determined by S	Section 3.3
$\hat{f}$	Scaled phase function	Section 3.4.1, Equation (3.9)
f_j	(Normalized) phase function lobe	Section 3.4.1, Equation (3.9)
W_j	Lobe weight	Section 3.4.1, Equation (3.9)
w_j	Lobe weight factor	Section 3.6.2, Equation (3.13)
$V(i)$	Set of input voxels contained in downsampled voxel i	Section 3.5
$V(i,j)$	The j-th cluster of $V(i)$	Section 3.5
$\mathrm{mean}(\cdot)$	Mean pixel intensity	Section 3.6
$\mathrm{mean}_k(\cdot)$	Restricted mean pixel intensity	Section 3.6

where σ_t (extinction coefficient), σ_s (scattering coefficient), and f (phase function) are *material scattering parameters*, and L is the resulting radiance field. In addition, *single-scattering albedo* is defined as the ratio between σ_s and σ_t: $\alpha = \sigma_s/\sigma_t$. Table 3.1 summarizes all symbols commonly used in this chapter.

Jakob et al. [42] generalized the RTE (3.1) to capture directional dependencies of the scattering parameters, yielding the *anisotropic* RTE

$$(\boldsymbol{\omega}\cdot\nabla)L(\boldsymbol{\omega}) = -\sigma_t(\boldsymbol{\omega})L(\boldsymbol{\omega}) + \sigma_s(\boldsymbol{\omega})\int_{\mathcal{S}^2} L(\boldsymbol{\omega}')f(\boldsymbol{\omega}'\to\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}'. \tag{3.2}$$

Under this formulation, the medium is modeled as a collection of two-sided small mirrors, or *micro-flakes*, whose normal directions $\boldsymbol{m}$ follow a statistical distribution D. Then, the scattering

parameters in Equation (3.2) are given by $\sigma_t(\boldsymbol{\omega}) = \rho\sigma(\boldsymbol{\omega})$, $\sigma_s(\boldsymbol{\omega}) = \alpha\sigma_t(\boldsymbol{\omega})$,

$$f(\boldsymbol{\omega}' \to \boldsymbol{\omega}) = \frac{1}{\sigma(\boldsymbol{\omega}')} \int_{\mathcal{S}^2} p(\boldsymbol{m}, \boldsymbol{\omega}' \to \boldsymbol{\omega}) \langle \boldsymbol{\omega}', \boldsymbol{m} \rangle D(\boldsymbol{m}) \, \mathrm{d}\boldsymbol{m}, \tag{3.3}$$

where ρ indicates the density of micro-flakes, α denotes the directionally independent albedo, p is the scattering profile for each micro-flake, and $\sigma(\boldsymbol{\omega}) = \int_{\mathcal{S}^2} \langle \boldsymbol{\omega}, \boldsymbol{\omega}' \rangle D(\boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}'$ provides the projected area of the micro-flakes in direction $\boldsymbol{\omega}$. This generalized framework collapses to the standard isotropic case (Equation (3.1)) when D is set to uniform [42]. In addition, it has been shown to be capable of closely capturing the appearance of complex materials such as fabrics and fur [29, 42, 118, 119].

Recently, Heitz et al. [29] introduced an SGGX-based representation for the micro-flake normal distribution:

$$D(\boldsymbol{m}) := \frac{1}{\pi\sqrt{|S|}(\boldsymbol{m}^T S^{-1} \boldsymbol{m})^2}, \tag{3.4}$$

where S is a 3×3 symmetric positive definite matrix. This distribution has been demonstrated to offer similar representative power as state-of-the-art models [42, 118] while being less expensive computationally. In the rest of this chapter, we use $f^{\mathrm{SGGX}}(S)$ to denote the SGGX phase function determined by S via Equations (3.3) and (3.4).

Interpolation of SGGX phase function. To represent the average of SGGX phase functions determined by matrices $S_1, S_2, \ldots, S_n$ with one SGGX function with the matrix $\tilde{S}$, Heitz et al. [29] demonstrated that using the arithmetic average of those matrices, namely setting

$$\tilde{S} = \frac{1}{n} \left(\sum_{i=1}^{n} S_i \right), \tag{3.5}$$

can lead to a good approximation.

Interpolation of density and albedo. Previously, Kraus and Bürger [51] demonstrated that independently interpolating (i.e., averaging) density and single-scattering albedo generally

yields very poor results. Instead, given n extinction coefficients $\sigma_{t,1}^{\text{orig}}, \ldots, \sigma_{t,n}^{\text{orig}}$ and corresponding albedo $\alpha_1^{\text{orig}}, \ldots, \alpha_n^{\text{orig}}$, it is better to set the interpolated extinction coefficient $\bar{\sigma}_t$ and albedo $\bar{\alpha}$ using:

$$\bar{\sigma}_t = \frac{1}{n}\sum_{i=1}^{n}\sigma_{t,i}^{\text{orig}} \quad \text{and} \quad \bar{\alpha} = \frac{\sum_{i=1}^{n}\sigma_{t,i}^{\text{orig}}\alpha_i^{\text{orig}}}{\sum_{i=1}^{n}\sigma_{t,i}^{\text{orig}}}. \tag{3.6}$$

This simple scheme, however, can still lead to limited accuracy. We use Equation (3.6) to create all naïve downsampling results.

3.4 Our Method

In this work, we aim to numerically compute downsampled representations of heterogeneous, anisotropic media with SGGX phase functions.

As demonstrated in prior work [50, 118], material appearance is insensitive to small changes of density σ_t (at a fixed resolution), and the relationship between the two is highly complex. Thus, we focus on computing albedos and phase functions with densities downsampled via Equation (3.6).[2]

3.4.1 Combining Albedo and Phase Function

Because our goal is to compute albedos α and phase functions f at reduced resolutions, it is desirable to have a framework combining these quantities. Given that $\sigma_s(\boldsymbol{\omega}) = \alpha\sigma_t(\boldsymbol{\omega})$, the anisotropic RTE (3.2) can be rewritten as

$$(\boldsymbol{\omega}\cdot\nabla)L(\boldsymbol{\omega}) = \sigma_t(\boldsymbol{\omega})\left[-L(\boldsymbol{\omega}) + \int_{\mathcal{S}^2} L(\boldsymbol{\omega}')\hat{f}(\boldsymbol{\omega}'\to\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}'\right], \tag{3.7}$$

where

$$\hat{f}(\boldsymbol{\omega}'\to\boldsymbol{\omega}) := \alpha f(\boldsymbol{\omega}'\to\boldsymbol{\omega}), \tag{3.8}$$

[2]It is also possible to compute and use downsampled albedos and phase functions using original (non-downsampled) densities. Our method is completely orthogonal and complementary to this aspect.

is a scaled version of the phase function f and is allowed to integrate to less than one. Our method focuses on numerically computing $\hat{f}$ at reduced resolutions. In particular, we treat $\hat{f}$ as a linear combination of m SGGX lobes. That is,

$$\hat{f}(\boldsymbol{\omega}' \to \boldsymbol{\omega}) = \sum_{j=1}^{m} W_j f_j(\boldsymbol{\omega}' \to \boldsymbol{\omega}), \tag{3.9}$$

where f_j is the j-th SGGX lobe and $W_j \in (0,1]$ denotes the weight of this lobe and satisfies $\sum_{j=1}^{m} W_j \leq 1$.

Given a scaled phase function (Equation (3.9)), the corresponding effective albedo α^{eff} and phase function f^{eff} can be obtained easily with

$$\alpha^{\text{eff}} = \sum_{j=1}^{m} W_j, \qquad f^{\text{eff}} = \frac{\hat{f}}{\alpha^{\text{eff}}}. \tag{3.10}$$

When $m = 1$, the lobe weight W_1 simply equals the effective albedo α^{eff} and $f^{\text{eff}} \equiv f_1$. When $m > 1$, each weight W_j affects both α^{eff} and f^{eff}.

3.4.2 Input and Output

Our approach takes voxelized representations of anisotropic media as input, where each voxel i stores:

- Optical density $\sigma_{t,i}^{\text{orig}}$, single-scattering albedo α_i^{orig};
- SGGX phase function f_i^{orig} determined by 3×3 symmetric positive definite matrix S_i^{orig}. Namely, $f_i^{\text{orig}} = f^{\text{SGGX}}(S_i^{\text{orig}})$.

The output of our approach is another volume with lower resolution, where each voxel i contains:

- Downsampled density $\bar{\sigma}_{t,i}$ given by Equation (3.6);
- Scaled phase function $\hat{f}_i$ determined by m SGGX lobes $f_{i,1},\ f_{i,2},\ \ldots,\ f_{i,m}$ (which in turn are given by matrices $S_{i,1},\ S_{i,2},\ \ldots,\ S_{i,m}$) and m weights $W_{i,1},\ W_{i,2},\ \ldots,\ W_{i,m} \in (0,1]$.

3.4.3 Overview

The rest of this chapter focuses on finding a proper scaled phase function $\hat{f}_i$ for each downsampled voxel i. That is, we need to determine the SGGX lobes $f_{i,j}$ (which is equivalent to finding $S_{i,j} \in \mathbb{R}^{3\times3}$) as well as the lobe weights $W_{i,j}$ for $1 \leq j \leq m$. Since there can be millions of voxels each of which has m unknown matrices $S_{i,j}$ and m unknown weights $W_{i,j}$, the problem is extremely under-constrained in general. To make it tractable, we start with determining $f_{i,j}$ locally based on constituent (input) phase functions (Section 3.5), and then optimize the weights $W_{i,j}$ globally (Section 3.6).

3.5 Determining SGGX Lobes

The m SGGX lobes $f_{i,1},\ f_{i,2},\ \ldots,\ f_{i,m}$ of a downsampled voxel i describe the *patterns* under which light scatters inside this voxel. Let $V(i)$ denote the set of input voxels (at the original resolution) that are contained in downsampled voxel i. Then, how light scatters is ultimately determined by the input phase functions $f_{i'}^{\text{orig}}$ with $i' \in V(i)$. We aim to find SGGX functions $f_{i,1}, f_{i,2}, \ldots, f_{i,m}$ capturing the *accumulated* scattering profile given by the input phase functions $f_{i'}^{\text{orig}}$.

The previous phase function downsampling approach [29] focuses on the special case of $m = 1$. In this case, $f_{i,1}$ needs to approximate $\sum_{i'\in V(i)} f_{i'}^{\text{orig}}$ and can be obtained directly via Equation (3.5). Namely, $f_{i,1} = f^{\text{SGGX}}(S_{i,1})$ with $S_{i,1} = \left(\sum_{i'\in V(i)} S_{i'}^{\text{orig}}\right)/n$.

Although using a single SGGX function to approximate the average of multiple SGGX functions works adequately in many cases (Figure 3.4, Example 1), it can lead to limited accuracy for highly anisotropic materials (Figure 3.4, Example 2).

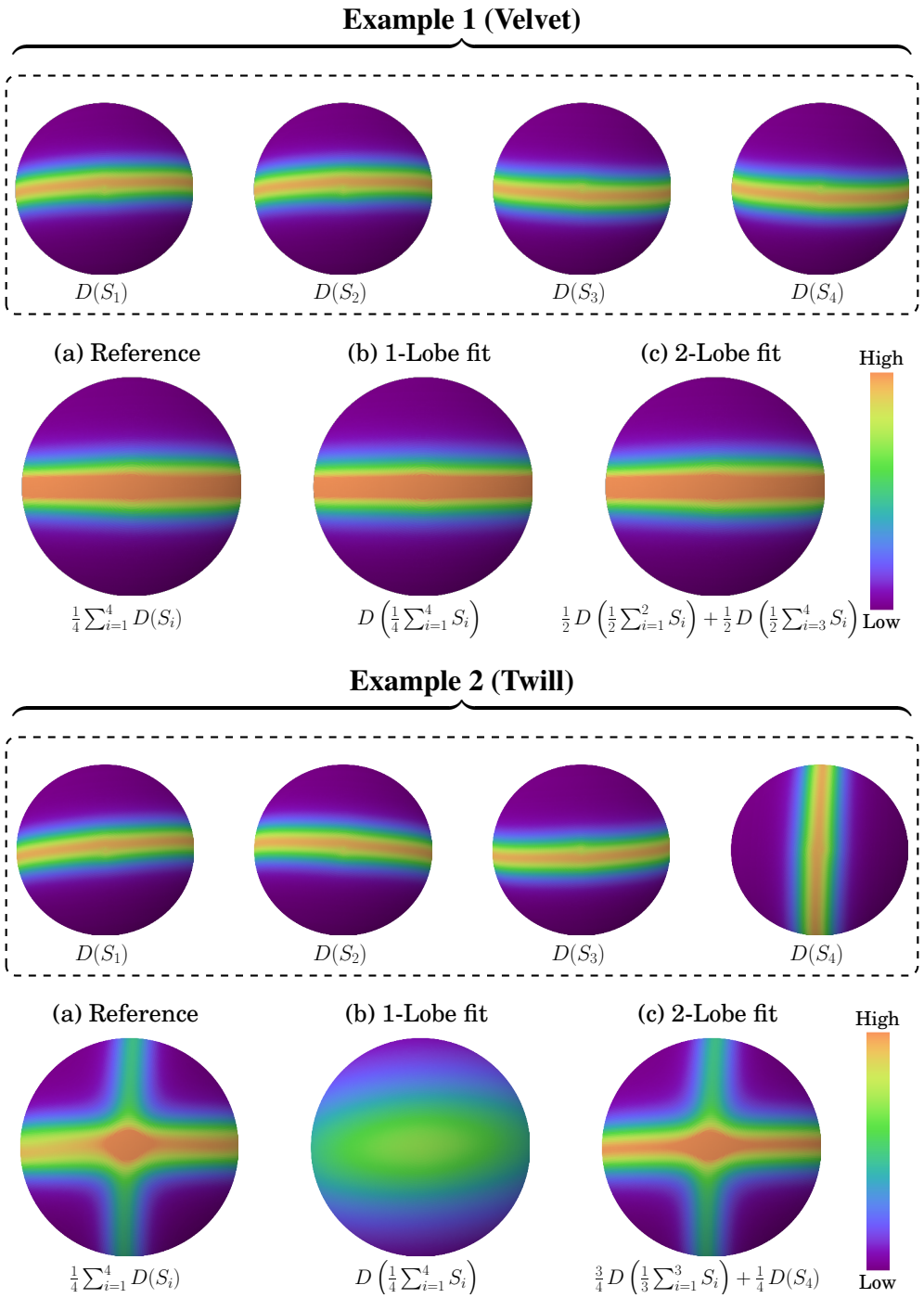


Figure 3.4. Comparison between single- and multi-lobe SGGX fits. In each example, the goal is to fit the reference (a) which equals the average of four SGGX lobes given by S_1, S_2, S_3, and S_4 (visualized in dashed box). Although the single-lobe fit (b) works well in Example 1, it fails to capture the high anisotropy in Example 2. The two-lobe fit (c), in contrast, works much better in this case. See Figure 3.17 for corresponding rendered images.

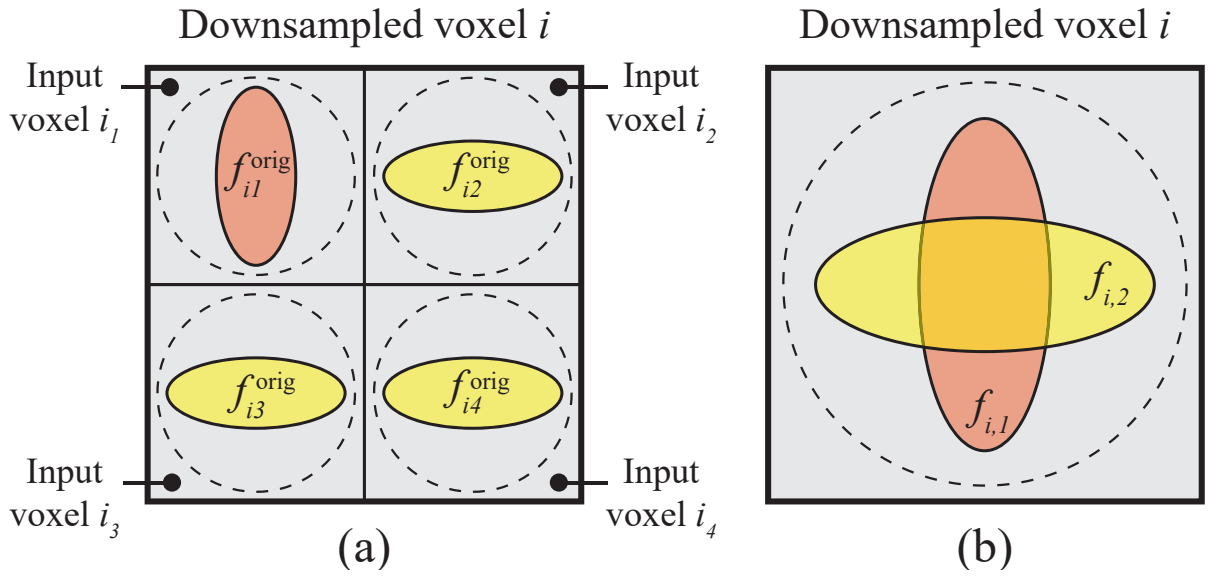


Figure 3.5. An example downsampled voxel i containing four input voxels (i.e., $V(i) = \{i_1, i_2, i_3, i_4\}$) is shown in (a), and a fitted two-lobe result in (b). The four input phase functions are partitioned into two clusters $V(i,1) = \{i_1\}$ (in orange) and $V(i,2) = \{i_2,\ i_3,\ i_4\}$ (in yellow) based on their shapes. The resulting two-lobe representation is then obtained by applying Equation (3.5) to each cluster. Note that this step only determines the shape of each lobe: their weights will be optimized globally in Section 3.6.

Phase Function Clustering. To allow $m > 1$, we partition the input phase functions $\{f_{i'}^{\text{orig}} : i' \in V(i)\}$ into m clusters and apply Equation (3.5) to each cluster. That is, we set

$$f_{i,j} = f^{\text{SGGX}}\left(\frac{1}{|V(i,j)|}\sum_{i' \in V(i,j)} S_{i'}^{\text{orig}}\right), \tag{3.11}$$

where $V(i,j) \subset V(i)$ denotes the set containing the indices of all input SGGX phase functions belonging to the j-th cluster. Figure 3.5 illustrates an example of this process. We compute the clustering $V(i,1),\ V(i,2),\ \ldots,\ V(i,m)$ using K-means with $S_{i'}^{\text{orig}}$ treated as 9D vectors.

To determine the value of m, one can start with $m = 1$ and iteratively increase m until the approximation error, i.e., the L_2 difference between $f_{i,j}$ and $\left(\sum_{i' \in V(i,j)} f_{i'}^{\text{orig}}\right)/n$, stops decreasing rapidly. In practice, we also limit m to 3 for a good balance between model size and result accuracy. See Section 3.8.1 for more details.

3.6 Optimizing Lobe Weights

With the SGGX lobes $f_{i,1},\ f_{i,2},\ ,\ldots,\ f_{i,m}$ determined at each downsampled voxel i, the remaining task for obtaining the scaled phase function $\hat{f}_i$ is to compute the corresponding *weight vector* $\boldsymbol{W}_i := (W_{i,1},\ W_{i,2},\ \ldots, W_{i,m})$.

Solving for m weights for each downsampled voxel is still a challenge due to the large number of unknowns. Fortunately, we have a good initial guess to start with: the (linearly) downsampled albedo $\bar{\alpha}_i$ which can be computed using Equation (3.6) for each voxel i. Since the scaled phase function is defined as the product of albedo and (normalized) phase function (Equation (3.9)), the naïve downsampling approach is equivalent to setting

$$W_{i,j}^{\text{naïve}} = \bar{\alpha}_i |V(i,j)|/|V(i)|, \tag{3.12}$$

where $|V(i,j)|/|V(i)|$ captures the weight of $f_{i,j}$ due to phase function clustering (see Equation (3.11) and Figure 3.5). In this case, it is easy to verify that $\alpha_i^{\text{eff}} = \sum_{j=1}^{m} W_{i,j}^{\text{naïve}} = \bar{\alpha}_i$.

3.6.1 Overview

The rest of this section focuses on finding *weight factors* for each downsampled voxel that can lead to good accuracy. These weight factors scale the lobe weights given by Equation (3.12). Because the number of downsampled voxels can be very large (in millions), solving for one set of weight factors per voxel is impractical. Instead, we partition the voxels into K clusters and search for one set of weight factors $\boldsymbol{w}_k := (w_{k,1},\ w_{k,2},\ \ldots,\ w_{k,m})$ for each cluster k. Then, for each downsampled voxel i, the lobe weights become

$$W_{i,j} = W_{i,j}^{\text{naïve}}\, w_{c(i),j} = \bar{\alpha}_i \frac{|V(i,j)|}{|V(i)|}\, w_{c(i),j}, \tag{3.13}$$

where $c(i) \in \{1,2,\ldots,K\}$ indicates the index of the cluster to which voxel i belongs.

We describe in Section 3.6.2 how to cluster the voxels and reorder the phase function

lobes $f_{i,j}$ so that each weight factor $w_{i,j}$ controls a set of lobes with similar shapes. Then, Section 3.6.3 introduces our main algorithm that numerically optimizes $\boldsymbol{w}_k$ for each voxel cluster k.

3.6.2 Voxel Clustering

We cluster the voxels by applying K-means to the downsampled albedos $\bar{\alpha}_i$. Our experiments indicate that a small number (e.g., two to five) of clusters can lead to high-quality results in practice.

When voxel cluster boundaries go across areas with smoothly varying albedos, the rendered images can occasionally contain visible seams. To ease this problem, we jitter the clusterings by slightly perturbing each $\bar{\alpha}_i$ when performing K-means. Please see Section 3.8.1 for examples.

Lobe ordering. Since each weight factor $\boldsymbol{w}_k$ is shared among multiple voxels (i.e., those with $c(i) = k$), the ordering of phase function lobes matters. Intuitively, we need to *reorder* the lobes $f_{i,1},\ f_{i,2},\ \ldots,\ f_{i,m}$ for each i so that those with the same indices from different voxels have similar shapes (see Figure 3.6).

For each voxel cluster, we pick an arbitrary voxel i as a reference and keep its lobe ordering fixed. Then, for each of the other voxels i', we search a permutation $\pi^*_{i'}$ of $\{1,2,\ldots,m\}$ such that the L_2 difference between corresponding SGGX normal distributions D (which determines actual phase function lobes) is minimized. Namely,

$$\pi^*_{i'} = \arg\min_{\pi} \sum_{j=1}^{m} \int_{\mathcal{S}^2} \left[D_{i,j}(\boldsymbol{m}) - D_{i',\pi(j)}(\boldsymbol{m}) \right]^2 \mathrm{d}\boldsymbol{m}. \tag{3.14}$$

We then use $\pi^*_{i'}$ to reorder the lobes of voxel i'. In practice, since the number of lobes m is usually very small, $\pi^*_{i'}$ can be computed in a brute-force manner.

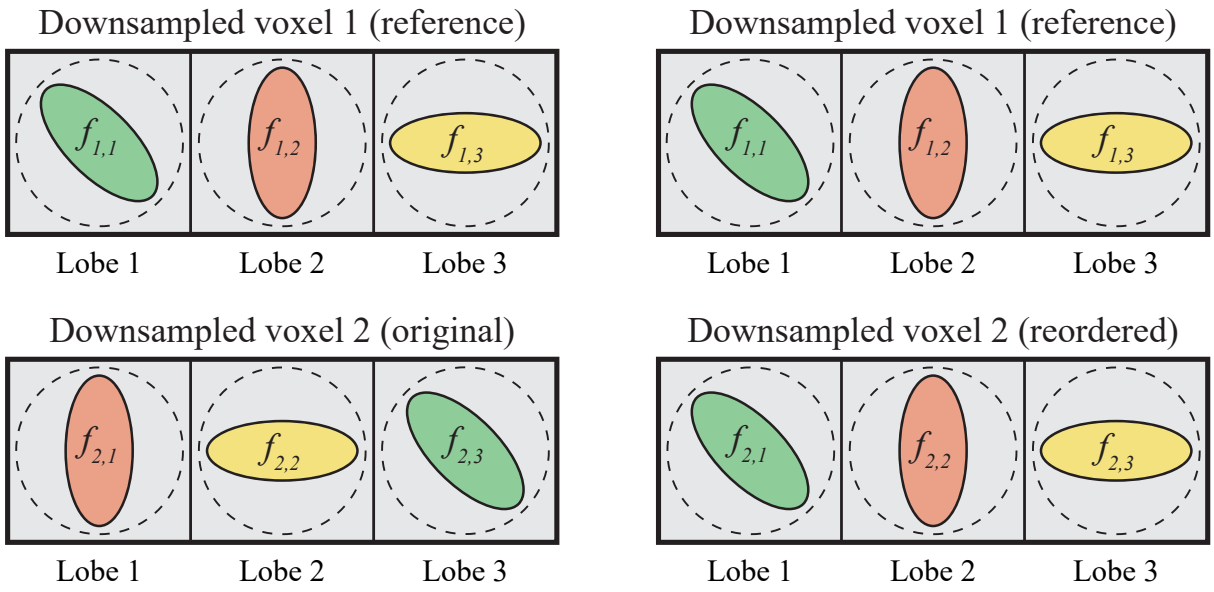


Figure 3.6. We **reorder** the phase function lobes so that those with identical indices have similar shapes. In this example, after reordering the lobes of voxel 2 using $\pi_2^* = (3,1,2)$, the shape of $f_{1,j}$ matches that of $f_{2,j}$ for $j = 1,2,3$.

3.6.3 Optimizing Weight Factors

We now present our method to search for proper weight factors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K$ (where K is the number of voxel clusters). Our goal is to make the resulting object appearance to closely match the ground truth under identical *smooth* lighting conditions.[3]

To obtain the optimal weight factors, we minimize an error metric based on differences between images rendered with original and downsampled parameters. Specifically, we perform training renderings with a number of lighting and viewing configurations, yielding a pair of images I_r (ground truth) and $\tilde{I}_r$ (approximated) for each configuration r. Notice that the latter image $\tilde{I}_r$ is a function of the weight factors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ that we are looking for.

Training scene setup. To generate the training renderings I and $\tilde{I}$, we need to design lighting and viewing configurations. For ensuring our solved albedo values generalize well to

[3]Here the smoothness assumption about lighting is a common practice when deriving approximate material properties and is used by the diffusion approximation [39] as well as similarity relations [120].

different settings, we perform the fitting using multiple lighting and viewing configurations. In particular, we render the object from the six directions (i.e., X-, X+, Y-, Y+, Z-, Z+) defined by its bounding box (Figure 3.7). For each view, we render the object using a few spherical harmonic lightings (see Section 3.8.1 for more details).

Error metric. To measure the difference between I_r and $\tilde{I}_r$ for each configuration r, we utilize a generalized version of the L_2 distance between mean pixel intensities. Given a rendered image I, let $\text{mean}(I)$ denote the average intensity of all (foreground) pixels in I. Previous work [50, 118] uses this measure to define their error metrics as

$$\sum_r c_r^2 \left(\text{mean}(\tilde{I}_r) - \text{mean}(I_r)\right)^2 \tag{3.15}$$

with $c_r = \max(\text{mean}(I_r),\ 0.05)^{-1}$ being a normalization factor. This metric works well when $\tilde{I}_r$ is affected by a small set of spatially invariant parameters, but has difficulties handling more general situations with parameter heterogeneity. In particular, the measure $\text{mean}(\tilde{I}_r)$ only tells if the whole image $\tilde{I}_r$ is too dark or too bright with little information on which voxel clusters are causing the problem, making it difficult to tell which weight factor needs to be corrected. Consequently, the optimization is prone to local minima.

We generalize $\text{mean}(I)$ and define the *restricted mean pixel intensity* $\text{mean}_k(I)$ as the average intensity evaluated among pixels to which downsampled voxels from cluster k are "directly" visible (see Figure 3.8). This leads to our error metric:

$$E_{\text{L2}}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) := \sum_r c_r^2 \sum_{k=1}^{K} \left(\text{mean}_k(\tilde{I}_r) - \text{mean}_k(I_r)\right)^2. \tag{3.16}$$

Our desired weights $\boldsymbol{w}_1^*,\ \ldots,\ \boldsymbol{w}_K^*$ minimize Equation (3.16). Namely,

$$(\boldsymbol{w}_1^*,\ \ldots,\ \boldsymbol{w}_K^*) = \underset{\boldsymbol{w}_1,\ \ldots,\ \boldsymbol{w}_K}{\text{argmin}}\ E_{\text{L2}}(\boldsymbol{w}_1,\ \ldots,\ \boldsymbol{w}_K). \tag{3.17}$$

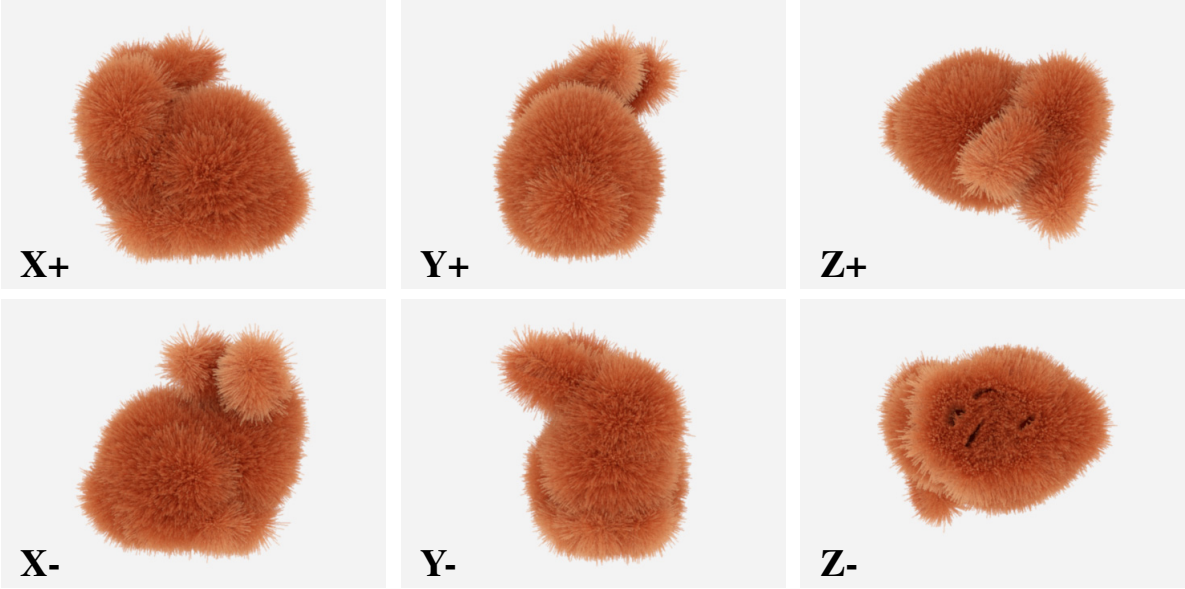


Figure 3.7. Six views we use for the training renderings.

We solve this optimization problem using *stochastic gradient descent* (SGD).

Gradient estimation. To solve Equation (3.17) using SGD, the key is to obtain an unbiased evaluation of the partial derivative of the error metric (Equation (3.16)) with respect to $w_{k,j}$ for any $1 \le k \le K$ and $1 \le j \le m$. It holds that

$$\begin{aligned} \frac{\partial E_{\mathrm{L2}}}{\partial w_{k,j}} &= 2 \sum_{r,k'} c_r \left[\mathrm{mean}_{k'}(\tilde{I}_r) - \mathrm{mean}_{k'}(I_r) \right] \frac{\partial \mathrm{mean}_{k'}(\tilde{I}_r)}{w_{k,j}} \\ &= 2 \sum_{r,k'} c_r \left[\mathrm{mean}_{k'}(\tilde{I}_r) - \mathrm{mean}_{k'}(I_r) \right] \mathrm{mean}_{k'} \left(\tilde{I}'_{r,k,j} \right), \end{aligned} \tag{3.18}$$

where $\tilde{I}'_{r,k,j}$ is the gradient image with respect to $w_{k,j}$ in which each pixel p has intensity $\tilde{I}'_{r,k,j}(p) = (\partial \tilde{I}_r / \partial w_{k,j})(p)$. We use path tracing to obtain an unbiased evaluation of $\tilde{I}'_{r,k,j}$. Please refer to Appendix A.1 for more details. Figure 3.9 shows an example of original and gradient images.

Stochastic gradient descent. With the gradient values, we can apply stochastic gradient descent (SGD) to find weight factors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ minimizing Equation (3.16). We use $\boldsymbol{w}_k^{(0)} = (1, \ldots, 1)$ for all $k = 1, 2, \ldots, K$ as the initial guesses. In each iteration, we update each weight

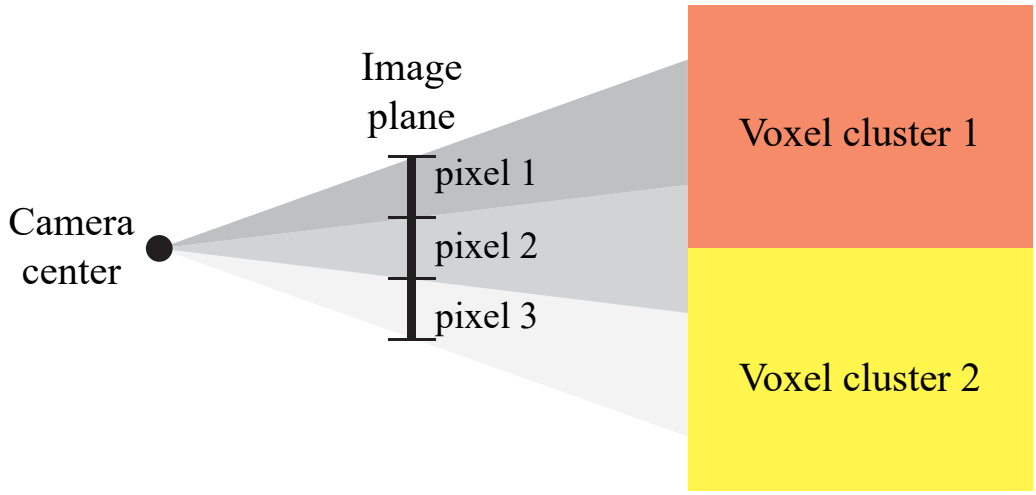


Figure 3.8. We define **restricted mean pixel intensity** that separates errors caused by different voxel clusters and yields better convergence of the optimization. In this 2D example has three pixels and two voxel clusters, $\text{mean}_1()$ is computed over pixels 1 and 2 because (at least part of) voxel cluster 1 is visible through these two pixels. Similarly, $\text{mean}_2()$ involves pixels 2 and 3.

factor $w_{k,j}$ via:

$$w_{k,j}^{(t+1)} = w_{k,j}^{(t)} - \delta_t \frac{\partial E_{L2}}{\partial w_{k,j}}\bigg|_{w_{k,j}^{(t)}}, \tag{3.19}$$

where δ_t is the step size of iteration t. The step size gradually decreases during iterations. In theory, δ_t needs to satisfy $\sum_{t=1}^{\infty} \delta_t = \infty$ and $\sum_{t=1}^{\infty} \delta_t^2 < \infty$ for ensuring convergence. We follow the common practice [50] of using the divergent harmonic series by setting

$$\delta_t = a_{k,j}/(t + b_{k,j}), \tag{3.20}$$

where $a_{k,j}$ and $b_{k,j}$ are constants. How to obtain the values for these constants is discussed in Section 3.8.1.

3.6.4 Handling Multiple Color Channels

All our derivations up to this point are for a single wavelength. In principle, when the input has colored scattering parameters, our downsampling pipeline needs to be executed per channel.

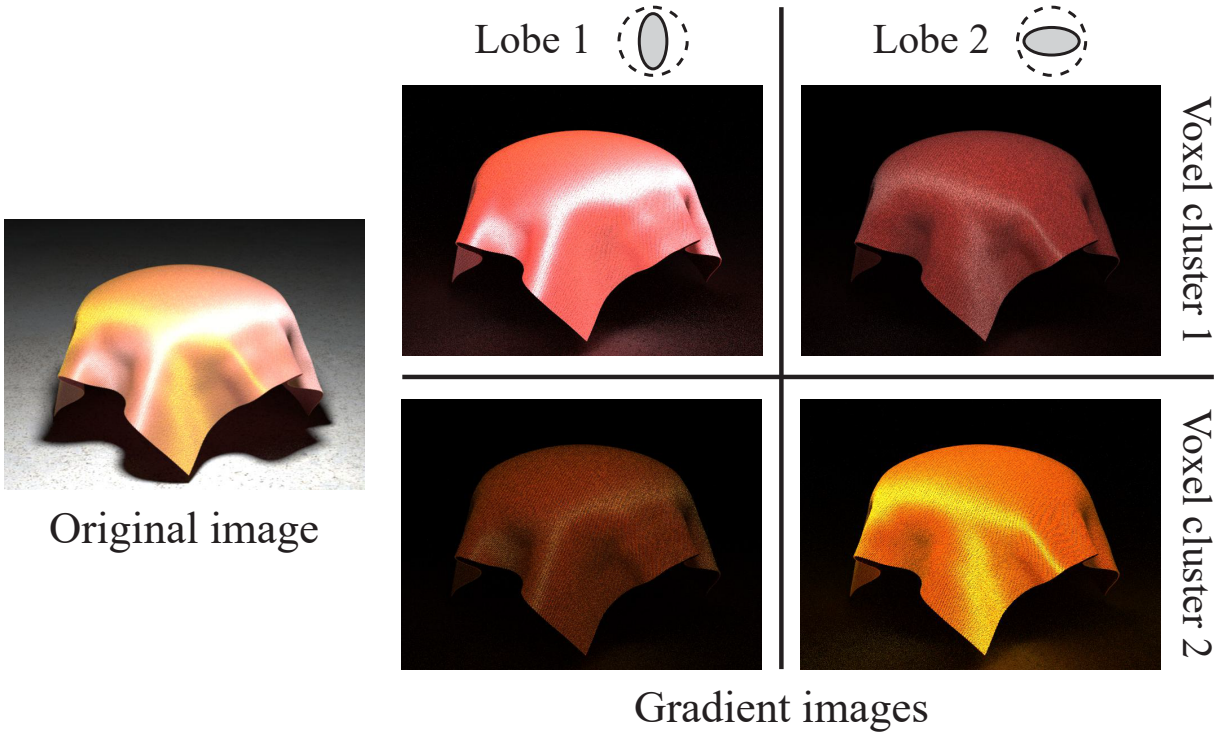


Figure 3.9. Original and gradient images rendered with our approach. This fabric contains differently colored warp (in pink) and weft (in yellow) yarns. Our method uses two voxel clusters (i.e., one for each type of yarn) and two phase function lobes per voxel. The corresponding gradient images successfully capture the main color of each cluster as well as anisotropy of the lobes. Furthermore, the varying average intensities of the gradient images demonstrate the correlation between voxel clusters and lobe weights: voxels in cluster 1 have high weights for lobe 1, and vice versa.

In practice, however, we found that the input parameters are usually *semi-colored*: only albedos vary between color channels while densities and phase functions stay constant. In this case, we keep SGGX lobes $f_{i,j}$ and weight factors $w_{i,j}$ single-channel and introduce an extra colored scaling factor $\boldsymbol{s}_k := (s_k^R, s_k^G, s_k^B)$ for each voxel cluster k, so that the lobe weights $W_{i,j}$ originally defined in Equation (3.13) become colored with

$$W_{i,j}^{\text{clr}} = s_{c(i)}\, \bar{\alpha}_i^{\text{clr}} \frac{|V(i,j)|}{|V(i)|} w_{c(i),j} \quad \text{for } \text{clr} \in \{R,G,B\}, \tag{3.21}$$

where $c(i)$ denotes the index of the cluster to which voxel i belongs, and $\bar{\alpha}_i^{\text{clr}}$ is the corresponding (i.e., red, green, or blue) component of the downsampled albedo at voxel i. We optimize

$\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K$ together with the weight factors $\boldsymbol{w}_{i,j}$ using SGD as this is more efficient than optimizing $\boldsymbol{w}_{i,j}$ for each color-channel separately (see Section 3.8.1 for an example).

3.6.5 Discussion

When $m = 1$ and $K = 1$, the scaled phase function at each voxel i becomes $\hat{f}_i = \bar{\alpha}_i w f_{i,1}$, and the optimization reduces to searching for a global weight factor w which effectively scales all albedos uniformly. Furthermore, if the input model has homogeneous albedo α, namely $\bar{\alpha}_i \equiv \alpha$ for all i, the search for w becomes equivalent to finding an altered albedo $\tilde{\alpha} := \alpha w$. This is precisely what previous methods [50, 118] do. Our method, on the other hand, is much more general and allows multiple lobes and voxel clusters as well as heterogeneous input albedos.

Voxel clustering. Our voxel clustering (Section 3.6.2) relies purely on downsampled albedo $\bar{\alpha}_i$ for each voxel i. This simple scheme has yielded high-quality results for all our experiments (see Section 3.8). In the future, more sophisticated features capturing spatial locations and/or phase functions can be used for handling input with higher degrees of correlations.

3.7 Exploiting Modularity

Many micro-appearance models consist of multiple *blocks* each of which comes form a predefined set of *exemplars*. For instance, Zhao et al. [119] introduced a *structure-aware synthesis* algorithm that automatically constructs highly complex fabric models based on exemplars with elementary weave patterns.

We exploit such modularity to accelerate the optimization of lobe weights. Our high-level idea is to directly downsample the exemplars (using techniques introduced in Sections 3.5 and 3.6) as precomputation. Then, to obtain a synthesized model at lower-resolution, we can directly replace its blocks with our downsampled versions. Because a synthesized model is normally much larger (in terms of the number of blocks) than the corresponding exemplars, it is more efficient to downsample the latter. More importantly, this allows many models synthesized from

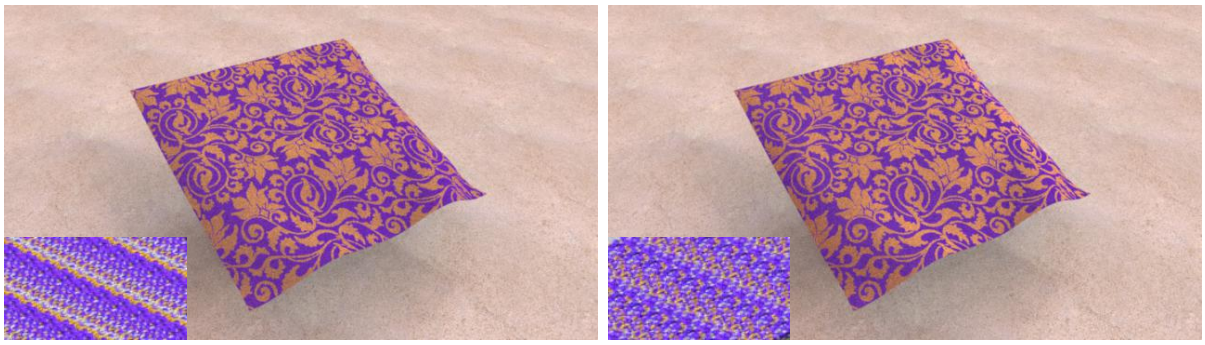


Figure 3.10. Two $(16\times)^3$-downsampled results synthesized with the same design but differently stacked exemplar volumes (whose tiled versions are shown as insets). The two exemplar stacking schemes have lead to visually identical results.

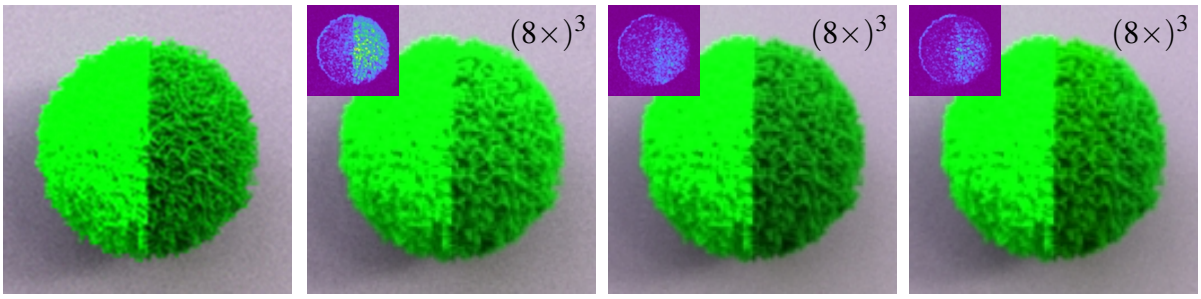
one set of exemplars to share one precomputation, greatly reducing the amortized downsampling overhead.

Downsampling exemplars. We downsample all exemplars together by stacking them (randomly) into a single *exemplar volume* and downsampling it using techniques introduced in Sections 3.5 and 3.6. Doing so has the following benefits. First, it guarantees downsampled voxels from different exemplars to have similar scaled phase functions if they contain similar input albedo and phase functions. This consistency is important for avoiding visible discontinuities in synthesized results. Second, it offers better performance than optimizing each exemplar separately.

In theory, because of high-order multiple scattering across neighboring exemplars, different ways of stacking can lead to varying optimization results. Fortunately, as demonstrated in Figure 3.10, we found that this hardly happens in practice: downsampled parameters using different stacking schemes generally yield visually identical results. Therefore, we compose the exemplar volume by stacking together individual exemplars in a randomized manner.

3.8 Results

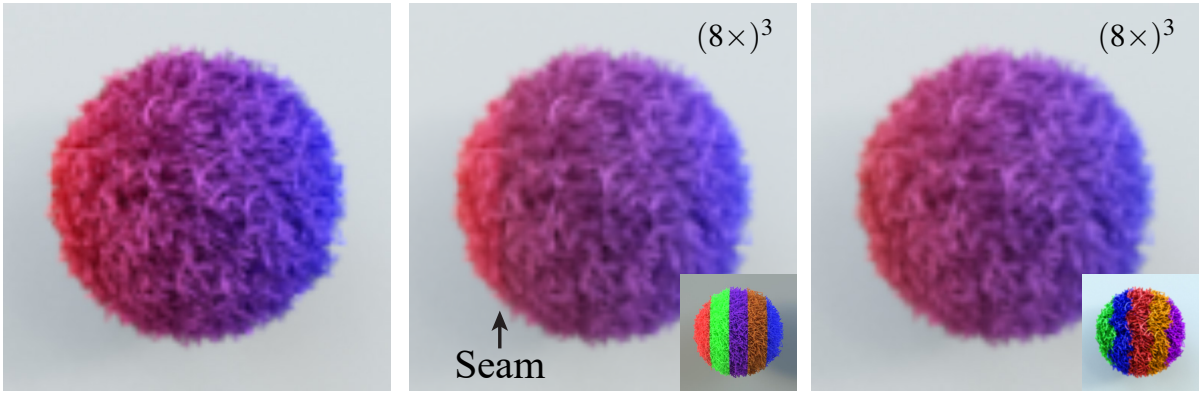
Using our technique described in Sections 3.5, 3.6 and 3.7, scaled phase functions (or, equivalently, albedos and phase functions) can be computed at greatly reduced resolutions. This



(a) Reference | **(b) 1 cluster** Rel. err: 0.105 | **(c) 2 clusters** Rel. err: 0.071 | **(d) 4 clusters** Rel. err: 0.067

Figure 3.11. Determining the number of voxel clusters. A hairy ball made with two materials (a). Each of these materials requires a distinctive albedo scaling to match the ground truth. If using only one cluster, the two materials are forced to be treated equally, resulting in poor accuracy (b). Using two clusters with our clustering scheme, each of the two materials will be handled separately, offering better accuracy (c). Going beyond two clusters (d), on the other hand, has limited benefits.

section shows results generated using our method. In Section 3.8.1, we empirically evaluate and justify several components of our approach. Then, in Section 3.8.2 we show downsampled results for a range of objects represented with high-resolution anisotropic volumes.



(a) Reference | (b) No jittering | (c) With jittering

Figure 3.12. Avoiding visible seams by jittering voxel clustering. When the input has smoothly changing albedos (a), neighboring voxel clusters can have clear boundaries that may yield visible seams (b). To ease this problem, we slightly jitter the clusters so that their boundaries become fuzzier (c).

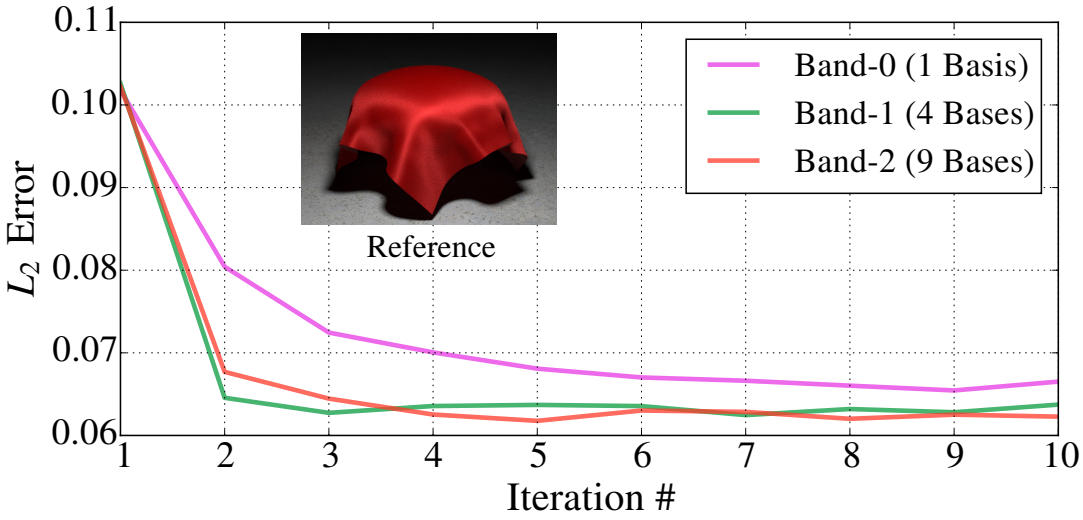
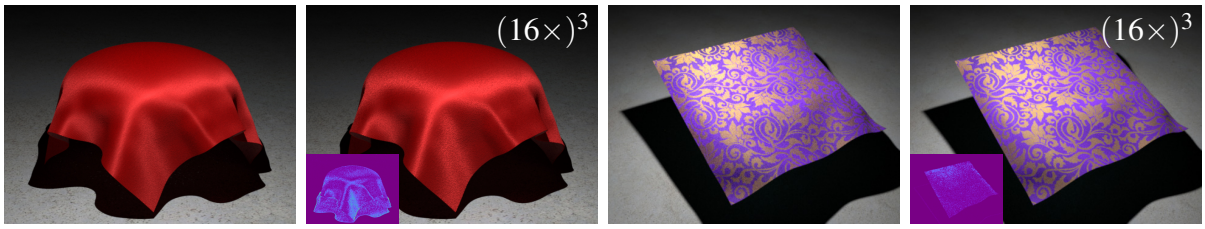


Figure 3.13. Selecting training lighting. The errors are evaluated between a reference image and those rendered using optimized parameters after a certain number of SGD iterations. An area light source is used to generate all these renderings. The results indicate that going beyond four SH bases offers little improvement for accuracy. The small differences between L_2 errors for band-1 and band-2 are mostly due to Monte Carlo noise.

3.8.1 Evaluations and Justifications

Voxel clustering. As described in Section 3.6.2, our method groups downsampled voxels into K clusters to make the optimization of weight factors tractable. In our experience, using one to five clusters generally provides a good balance between accuracy and performance (see Figure 3.11). We also jitter the voxel clustering for input with continuously changing albedos to ease potential seams (Figure 3.12).

Lighting in training renderings. We use four SH lightings (i.e., first two bands) for our training renderings. Figure 3.13 shows an example where we optimized the homogeneous twill (Figure 3.3) using one, four, and nine SH lightings (corresponding to band-zero, one, and two) and the six views shown in Figure 3.7. The L_2 errors are computed between the reference image and ones rendered using parameters obtained after a certain number of SGD iterations. The results indicate that using more than four SH bases yields little benefit in terms of accuracy. Thus, we use four SH lightings for our training renderings.

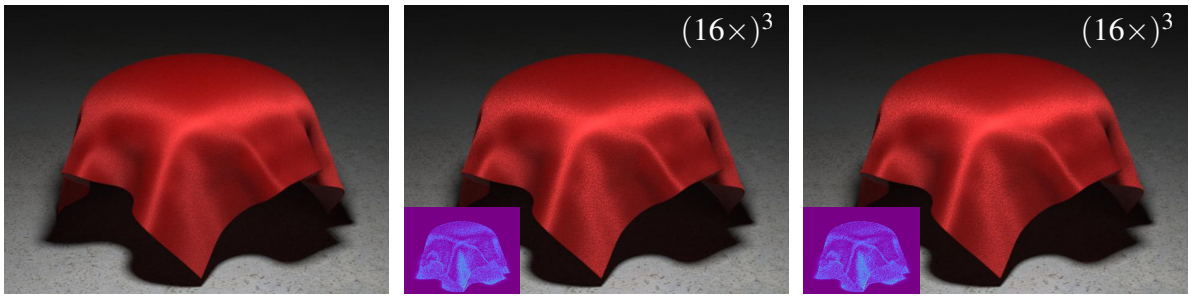


(a1) Reference (b1) Ours (a2) Reference (b2) Ours

Figure 3.14. Although our training renderings use smooth SH lightings, the optimized parameters generalize well to harsh lighting conditions. These two examples use local point light sources which lead to sharp shadow boundaries on the ground.

Harsh lighting. We use SH lighting for the training renderings to allow the optimized parameters to generalize well to arbitrary smooth lighting conditions. In theory, when these parameters are rendered under harsh lighting, the resulting accuracy can degrade. However, we did not observe such cases in practice: our optimized parameters generalize well even to extremely high-frequency lightings (see Figure 3.14 for two examples).

Handling multiple color channels. As described in Section 3.6.4, we optimize one extra term $\mathbf{s}_k$ for each voxel cluster k to handle *semi-colored* input where only single-scattering albedo varies between different color channels. As shown in Figure 3.15, this is more efficient than optimizing $\boldsymbol{w}_{i,j}$ for each color-channel separately while providing similar accuracy.



(a) Reference (b) Joint opt. (c) Independent opt.

Figure 3.15. For semi-colored input, optimizing multiple color channels jointly (b) is more efficient than optimizing each channel independently (c) while yielding similar resulting accuracy.

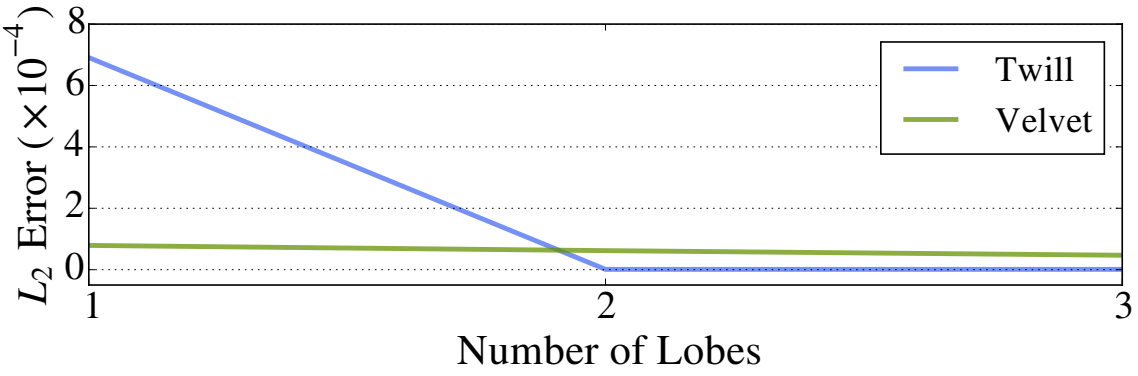


Figure 3.16. Approximation error (measured in L_2) decreases as the number of phase function lobes increases.

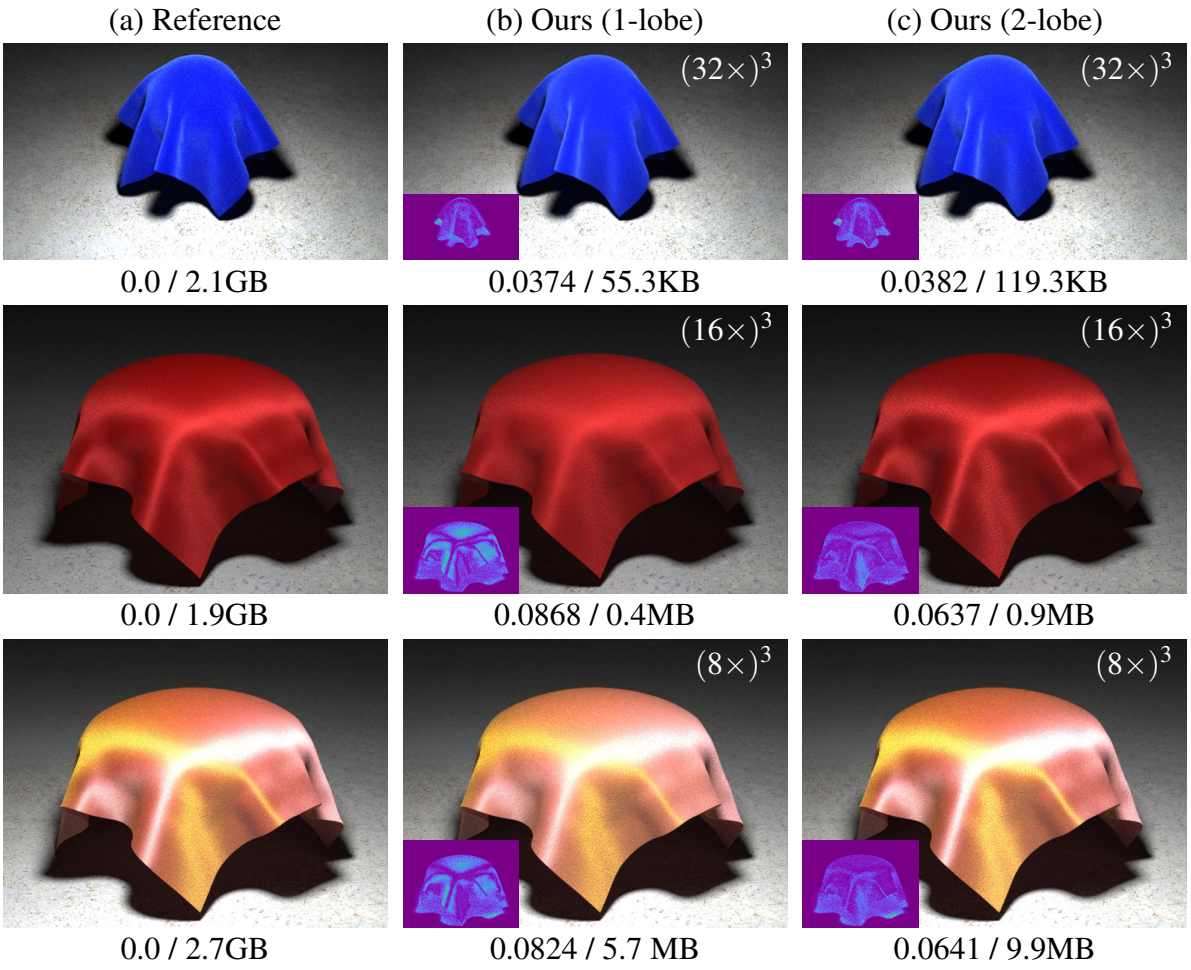


Figure 3.17. Optimized results using **varying numbers of phase function lobes**. Relative error and data size are shown below each image. For each example, optimizations corresponding to (b) and (c) take similar time to converge. For the velvet (the top row), one lobe suffices. For the twill (the bottom two rows) with high anisotropy, on the other hand, using two lobes has led to superior accuracy.

Table 3.2. Stochastic gradient descent settings (i.e., number of voxel clusters K and number of phase function lobes per voxel m) and optimization time (in CPU core hours) for all results in Figures 3.18 and 3.19. Our optimization time is comparable to the rendering time (shown in the last column) for one image at 720p.

Object	m	K	# Iter.	Opt. time	Render time
Bunny	1	1	8	12	8
Twill (homogeneous)	2	1	20	33	23
Velvet	1	1	15	35	74
Hairy ball	1	5	20	40	30
Twill (heterogeneous)	2	2	25	45	30
Damask	3	3	30	60	51

Number of phase function lobes. As discussed in Section 3.5, we determine the number of phase function lobes m by iteratively increasing m until the approximation error stops descending rapidly. In particular, we stop increasing m when the L_2 error with $(m+1)$ lobes is greater than 50% of that with m lobes. Figure 3.16 shows how the approximation error changes with the number of lobes for the example from Figure 3.4. Based on the aforementioned scheme, we use two lobes for the twill while one for the velvet. Figure 3.17 contains the corresponding renderings justifying our choices of lobe counts. Further, we limit m to 3 in all our experiments for a good balance between model size and result accuracy. Lastly, notice that when using only one lobe (i.e., $m = 1$), our method effectively optimizes only the downsampled albedos $\tilde{\alpha}_i$.

3.8.2 Main Results

We evaluate the effectiveness of our approach using six examples. We tune $a_{k,j}$ and $b_{k,j}$ in Equation (3.20) manually using a subset of lighting/viewing conditions so that the error metric (Equation (3.16)) changes neither too quickly (resulting in oscillation) nor too slowly (leading to slow convergence). The overhead for this extra tuning is less than 10% of the total optimization time. Since we use low-resolution (around 100p) and noisy training renderings, the total optimization time for each object is comparable to the rendering time for one image at

720p. Our optimization configurations and performance numbers are summarized in Table 3.2. The values of all step sizes $a_{k,j}$, $b_{k,j}$ as well as optimized weight factors $w_{k,j}$ and colored scaling factors $\mathbf{s}_k$ are available as supplemental material of the original publication [121]. We use a modified version of the Mitsuba renderer [41] to generate all regular and gradient images.

Figure 3.18 shows lower-resolution volumes generated with our approach. The first three examples have homogeneous albedos while the last two have heterogeneous ones. Each object is rendered under two environmental lightings that are significantly more complicated than our training ones (i.e., four SH lightings).

The first row of Figure 3.18 contains a hairy bunny. After downsampling at $(4\times)^3$, our optimized parameters using one lobe per voxel generalize well to different environmental lightings. The second row of Figure 3.18 shows a shiny twill fabric originally represented with 4.76×10^{11} effective voxels. Because of the highly anisotropic nature of this material, we use two lobes per voxel. Our result, which contains only 1.07×10^8 effective voxels, achieves more than three orders of magnitude storage saving and successfully preserves this material's glossy appearance under both local (left) and global (right) lighting. The third row of Figure 3.18 contains a highly scattering velvet with 7.77×10^{11} effective voxels. The velvet's highly detailed surface structure caused naïve downsampling to have very poor accuracy. Our model computed with one lobe per pixel, on the other hand, has 2.03×10^7 effective voxels and maintains good accuracy while providing four orders of magnitude data reduction. The quality of our optimized parameters for this model is further demonstrated in the supplemental video of the original publication [121]. In the fourth row, we show a hairy ball with gradually changing colors. Our optimization uses five voxel clusters with jittering (visualized in the insets) and one lobe per voxel. The result achieves good accuracy while reducing the storage by almost three orders of magnitude. The bottom row of Figure 3.18 contains another complex twill scene with differently colored warps and wefts (identical to Figure 3.9). This heterogeneity results in distinctive dual-colored anisotropic highlights. Our result based on two voxel clusters and two-lobe phase functions maintains the appearance of this object well even at the yarn-level.

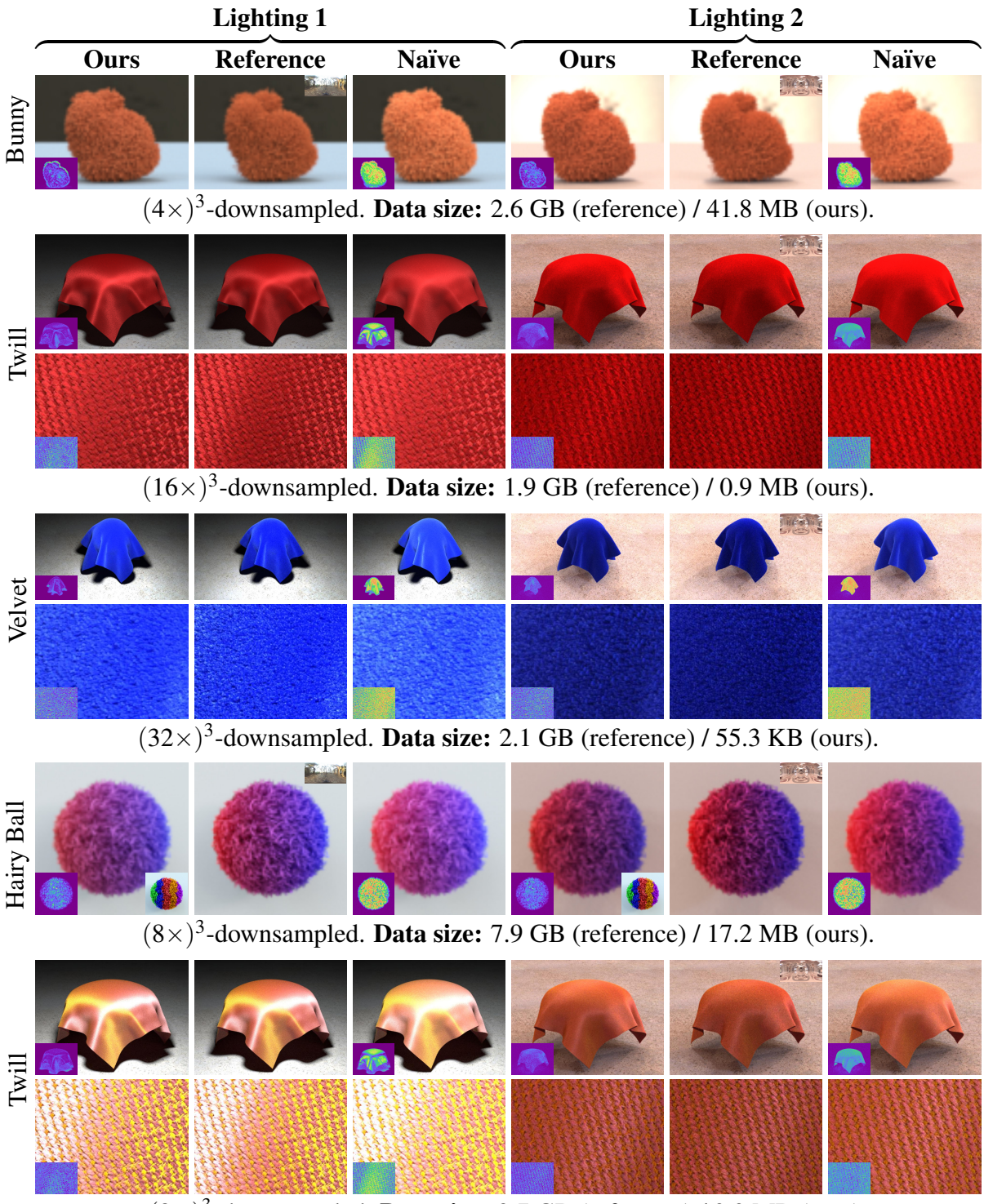


Figure 3.18. Main results. Our reduced-resolution representations provide good accuracy under general lighting with up to four orders of magnitude storage saving. Environmental lightings used in these renderings are visualized in the top-right corner of individual reference images. Those without lighting visualizations use local area light sources. Please refer to Table 3.2 for optimization and rendering times.

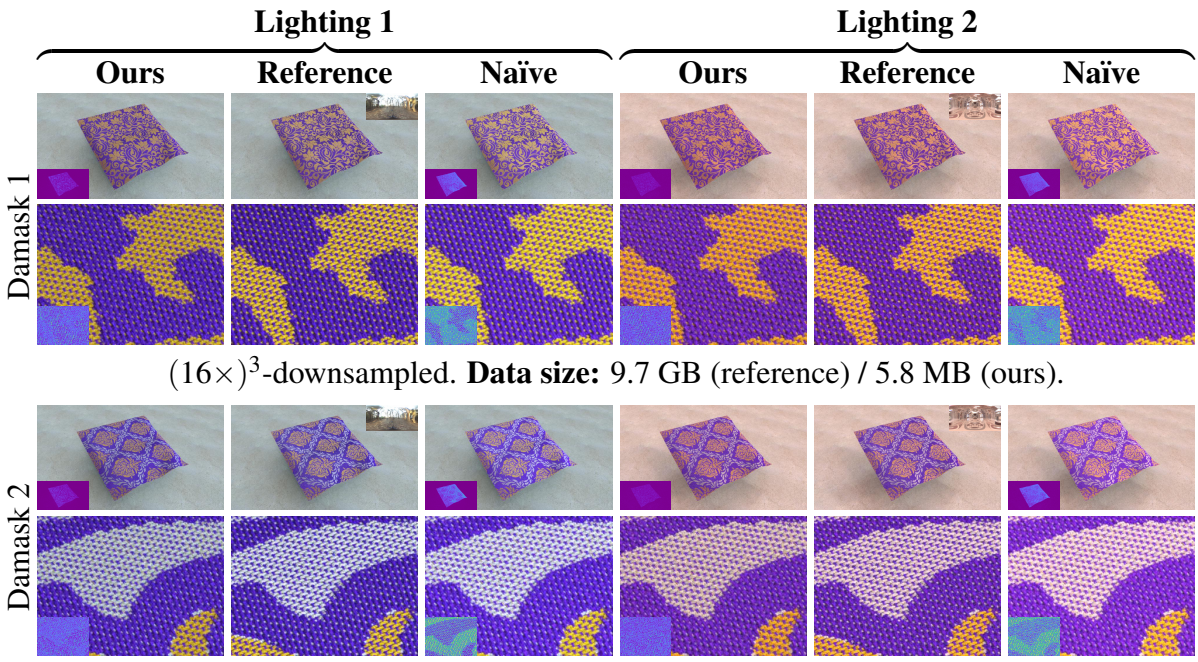


Figure 3.19. Exploiting modularity. The two downsampled damask fabrics sharing one optimization offer good accuracy and three orders of magnitude storage saving. The corresponding environmental lightings are shown in top-right corners of the reference images. Please refer to Table 3.2 for optimization and rendering times.

Figure 3.19 shows downsampled results from two damask fabrics synthesized from a single set of 120 example blocks. We downsample the exemplars using three voxel clusters and three lobes per voxel. This precomputation takes 60 core hours. Then, the two downsampled damasks are obtained by stacking the pre-downsampled example blocks. Based on the single precomputation, our approach results in good accuracy while reducing the amount of data by three orders of magnitude.

Limitations and future work. Our method is based on optimization and does not explicitly reason about how downsampling scattering parameters affects light transport in participating media. A theoretical analysis on this topic would be valuable and may inspire future improvements to our optimization framework.

3.9 Conclusion

We have introduced an optimization-based approach to compute scaled phase functions, a combined representation of single-scattering albedo and phase function, for downsampling voxelized anisotropic media. Our method starts with determining phase function lobes locally by clustering input phase functions. Then, we optimize lobe weight factors globally via stochastic gradient descent. The resulting representation can offer several orders of magnitude reduction in storage while maintaining good accuracy. In addition, we demonstrated that modularity can be exploited for synthesized models to greatly reduce amortized downsampling overhead.

3.10 Acknowledgements

This chapter is based on the material as it appears in ACM Transactions on Graphics, 2016 ("Downsampling Scattering Parameters for Rendering Anisotropic Media", Shuang Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 4

Accurate Appearance Preserving Prefiltering for Rendering Displacement-Mapped Surfaces

In addition to volumetric scattering models, complex materials can also be modeled by detailed surfaces. In this chapter, we demonstrate a new approach to prefilter opaque surfaces whose micro-scale geometric details are represented by high-resolution displacement maps. To accurately match the appearance, we extend the effective BRDF to capture the interreflection effect within the micro-geometry, and use it as an indicator of appearance accuracy. This avoids the expensive iterative optimization used in Chapter 3, allowing efficient prefiltering. The key component of our method is a novel scaling function that captures micro-scale changes of shadowing, masking, and interreflection effects. Our method generalizes well to different types of micro-geometries beyond Gaussian or GGX surfaces.

4.1 Introduction

High-resolution displacement maps are commonly used to describe detailed micro-geometries that can produce richly diverse appearances. Compared to normal mapping, displacement mapping is more physically consistent and can offer more realistic self-shadowing and silhouettes. However, such realism comes at the cost of difficult prefiltering: smoothing a displacement map usually weakens its intrinsic shadowing and results in brightened overall

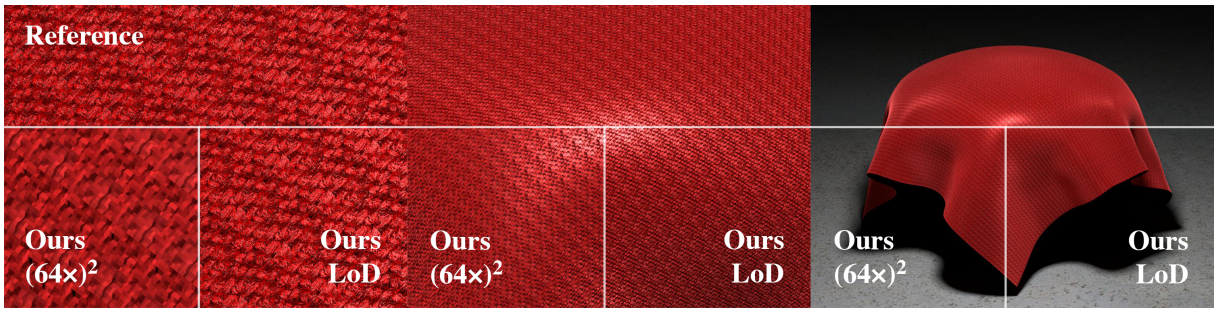
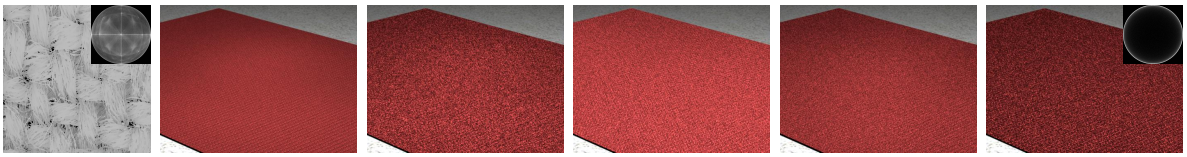


Figure 4.1. We present a new approach to prefilter high-resolution displacement maps while preserving the input appearance. High-resolution displacement maps can produce rich geometric details (top-left) but they are difficult to prefilter. Our prefiltered model handles the change of shadowing, masking and interreflections caused by downsampling the displacement map. At a single scale, although the detailed micro-structures are different (see ours $(64\times)^2$ at left), our prefiltered model preserves the original appearance accurately when we view the object from a distance (see the right image). We can also combine our models at multiple downsampling scales to form a mipmap, enabling accurate and anti-aliased LoD rendering.

appearance. Therefore, rendering a high-resolution displacement map without introducing severe aliasing generally requires significant super-sampling, which is computationally expensive.

Previous displacement mapping techniques such as LEAN [77] and LEADR [17] can produce anti-aliased renderings of rough surfaces. However, they assume the normals of the input surfaces to have Beckmann distributions, which is usually violated in practice and fundamentally limits the accuracy of these methods (Figure 4.2(f)). To handle a wider range of surfaces, some bi-scale appearance models [40, 107] precompute the overall surface reflectance by averaging light reflections at the micro-scale. This yields one spatially macro-scale effective BRDF with no spatial variation, which has difficulties in reproducing micro-scale details at close-up views. Both approaches neglect interreflection and can lead to significant energy loss.

In this chapter, we introduce a novel method to prefilter displacement-mapped (opaque) surfaces while accurately preserving their overall appearances under both direct and global illuminations (e.g., Figures 4.1 and 4.2). Given a high-resolution displacement map and an isotropic base micro-BRDF, we seek displacement maps and accompanying surface reflectance models that have reduced resolutions and closely preserve the macro-scale appearance of the input model. To this end, we leverage a generalized version of the effective BRDF formulation [107]



(a) Disp. map | (b) Reference 1000 spp, 500s | (c) Original 50 spp, 26s | (d) Naïve 80 spp, 27s | (e) Ours 35 spp, 26s | (f) LEADR 70 spp, 26s

Figure 4.2. Equal-time renderings with only direct illumination. Given a high-resolution displacement map (a) and a highly glossy base BRDF as input, directly rendering using them results in aliased results (c). Simply downsampling the displacement map to a lower resolution but keeping the original base BRDF leads to inaccurate appearance (d). Our method (e) prefilters the displacement map and the base BRDF while preserving the input appearance (b), which produces accurate and anti-aliased renderings. Using the microfacet BRDF proposed in LEADR [17] generates inaccurate appearance (f) since the Gaussian surface assumption is violated. We show the ground truth NDF visualization of this displacement map in the top-right inset of (a), and the NDF estimated from LEADR in the top-right inset of (f), which is very different.

that takes both shadowing-masking and interreflection effects into account. Based on this formulation, we directly match the effective BRDFs of the input model and our prefiltered variants.

Our method (Figure 4.3) starts with downsampling the input displacement map. This is achieved via an optimization that aims to preserve the surface's meso-scale geometries (e.g., normals) and minimize potential energy loss. The resulting surface, when coupled with the input reflectance model, generally leads to a different appearance (see Figure 4.2(d) for an example) due to the change of shadowing structures, distribution of normals, and interreflections caused by the downsampling. The main focus of this chapter, therefore, is to find proper surface reflectance models (i.e., BRDFs) so that, when coupled with the downsampled geometries, they closely resemble the appearance of the input.

To fully preserve the detailed appearance of the input model using the downsampled geometry, a 6D spatially varying effective BRDF is generally required. Unfortunately, explicitly expressing this SVBRDF requires significant computation and storage, negating the benefits of displacement map downsampling. Instead, we decompose the 6D function into a 4D spatially varying normal distribution function (SVNDF) [25] and a novel scaling function. The SVNDF,

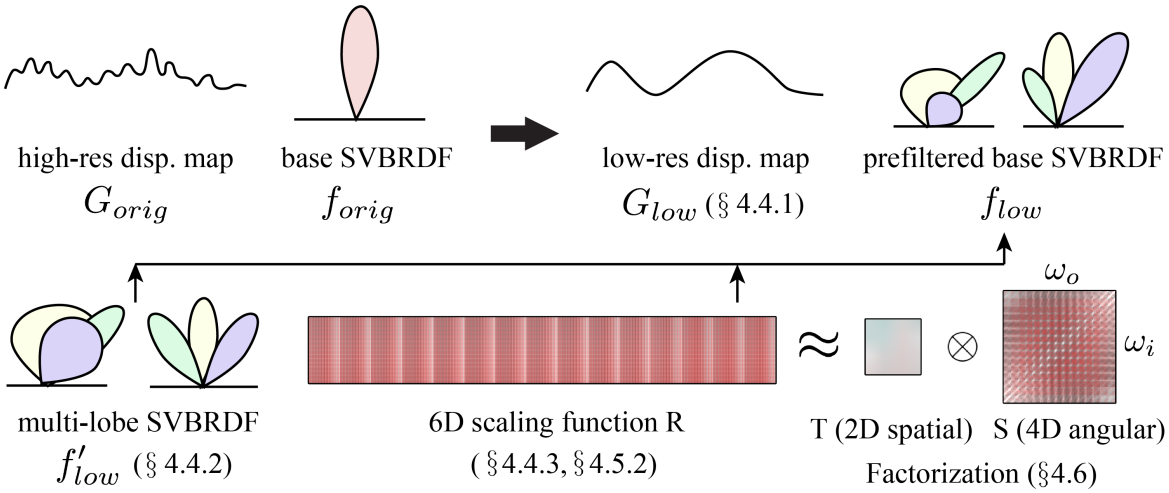


Figure 4.3. Workflow of our method. We convert a high-resolution displacement map and base BRDF (top-left) to a low-resolution one and prefiltered SVBRDF (top-right). The prefiltered SVBRDF is obtained as a product of the SVBRDF from a multi-lobe NDF (bottom-left) and a 6D scaling function (bottom-right). This 6D function can be further factorized as the product of a single 2D spatial scaling function and a single 4D angular scaling function.

which neglects both shadowing-masking and interreflection, primarily encodes spatial variations of the effective BRDFs and can be represented as 2D spatially varying parametric distributions. The scaling function, in contrast, accounts for the shadowing-masking and interreflection effects and is vital for matching the input appearance. To efficiently describe this scaling function, which is 6D itself, we factorize it into a 2D spatial scaling function of surface location and a 4D angular scaling function of direction (e.g., Figure 4.10). Further, we exploit the fact that the scaling functions are usually low-frequency and smooth in practice to introduce a simple and efficient factorization method to compute them from a sparse set of samples. With a resolution up to 4^2 for the spatial function and 15^4 for the angular one, our scaling functions only consume 200–400 KB of storage, making our prefiltered models compact and practically useful. Our contributions include:

- We develop a novel method to prefilter the surface reflectance of a high-resolution displacement map while accurately preserving the overall appearance (Figures 4.1, 4.16, 4.17, 4.18). This is achieved by first downsampling the displacement map to minimize the meso-scale average

Table 4.1. We compare our method with LEADR [17], bi-scale appearance models [40, 107] and multi-scattering microfacet models [31, 59, 110]. Our method can handle both shadowing-masking and interreflections but is not limited to Gaussian/GGX surfaces, at the expense of some precomputation.

Method	LEADR	bi-scale	microfacet	ours
Shadowing and masking	✓	✓	✓	✓
Interreflections	✗	✗	✓	✓
General surfaces	✗	✓	✗	✓
No Precomputation	✓	✗	✓	✗

surface slopes (Section 4.4.1), then separating a 6D SVBRDF into a SVNDF (Section 4.4.2) and a scaling function (Section 4.4.3).

- To match the target appearance, we utilize two scaling functions. One captures only shadowing-masking (Section 4.4.3) and the other effectively handles both shadowing-masking and interreflection (Section 4.5.2). To model interreflection within the micro-geometry, which is generally missing in previous work, we introduce a generalized formulation of effective BRDFs (Section 4.5.1).
- We show the scaling function can be factorized as the product of a single spatial scaling function and a single angular scaling function. Exploiting their low-frequency property and smoothness, we present a simple and efficient method to compute the spatial and angular scaling functions (Section 4.6).
- To enable level-of-detail (LoD) rendering, we present a linear interpolation method that provides smooth transitions between our models prefiltered at varying scales (Section 4.7.2).

4.2 Related Work

We review several main research areas in the following paragraphs. Comparison between our method and the most related techniques is summarized in Table 4.1.

Surface appearance prefiltering. We refer readers to the survey of Bruneton and

Neyret [7] for a comprehensive review of surface appearance prefiltering techniques. Traditional normal/BRDF map filtering methods [18, 25, 49, 95, 97, 111] ignore shadowing and masking effects. Tan et al. [94] approximate shadowing and masking using horizon map distributions. LEADR [17] extends from LEAN mapping [77] by incorporating a physically based shadowing-masking term derived from microfacet theory. In LEADR, they use one-lobe NDFs for simplicity and efficiency in a real-time rendering context. We use multi-lobe NDFs to achieve better accuracy, especially at glossy highlights (Figure 4.16). Recently, Loubet and Neyret [66] propose a hybrid mesh-volume model to construct LoDs of complex objects. They leverage volumetric models to handle self-occlusions caused by micro-geometries, while our method generates pure surface reflectance models that are simpler and more efficient to use. In addition, none of these methods take interreflections into account, since interreflections combined with shadowing and masking are complicated to analyze and prefilter.

Bidirectional texture functions (BTF)[8, 16] capture spatially varying and view-dependent surface appearance. Generating 6D BTF data requires expensive precomputation and a large amount of storage. Although BTF filtering methods [44, 67, 106] provide a direct solution to surface appearance prefiltering, the high dimensionality limits their practical applications. Our method separates a full 6D function into a 4D SVNDF and a 6D scaling function. The latter scaling function can be further factorized into a single 2D spatial scaling function and a single 4D angular scaling function, which is cheaper to compute and compact to store. Complex shadowing-masking and interreflections can be captured by the spatial and angular scaling functions accurately.

Bi-scale material design. A series of works model object's macro-scale appearance by manipulating its micro-scale details [27, 40, 105, 107]. Westin et al. [105] explicitly simulate ray tracing on micro-geometry. Heidrich et al. [27] speed up the simulation on height fields using precomputed visibility. Wu et al. [107] propose an interactive bi-scale material editing system. They precompute rotated BRDF values and bidirectional visible normal distribution functions

(BVNDF) and compress them as low-rank matrices. This work has been extended to edit highly glossy materials by using mixtures of spherical Gaussians (SG) or anisotropic spherical Gaussians (ASG) [40, 112]. The shadowing-masking term is controlled by the weights of SG/ASG lobes. Most of these bi-scale appearance modeling techniques focus on the average large-scale appearance and do not consider spatial variation. In addition, the interactive approaches have no interreflection components. Recently, several methods for rendering glinty surfaces have been developed by simulating specular reflection on micro-surfaces [12, 43, 113, 114]. Zirr and Kaplanyan [123] propose a bi-scale microfacet model to render micro-details in real-time. Their methods focus on spatially varying NDFs, neglecting shadowing-masking and interreflections.

Microfacet models. Microfacet models describe the aggregate reflectance from a statistical representation of rough surfaces, i.e., the orientations of microfacets, resulting in a number of physically based BRDFs [15, 78, 101]. The Smith model [28, 90] gives an accurate approximation of the microfacet shadowing-masking function with the assumption of independence between heights and normals. In particular, the shadowing-masking function has analytic solutions if Gaussian or GGX surfaces are given. Ashikmin et al. [4] derive a 4D BRDF from a 2D NDF, in which the shadowing-masking term is numerically computed from the NDF. LEADR [17] assumes Gaussian surfaces and leverages the Smith model to handle masking and shadowing effects. Recent works [31, 59, 110] extend microfacet theory by modeling multiple scattering inside the micro-surface. Though microfacet models enable efficient computation of shadowing-masking and interreflections, they are limited by Gaussian surfaces, GGX surfaces or V-grooves. Our method can handle general surfaces without assumptions for specific micro-geometries.

Inverse rendering. Inverse rendering optimizes for scene parameters that produce the best match to target appearance. Previous methods [21, 22, 26, 50, 121] require global inverse rendering, which solves for a number of scattering parameters and involves expensive Monte Carlo path tracing during iterative optimizations. We just need to perform standard normal

mapping and precompute simpler effective BRDFs once to obtain the prefiltered scattering parameters. Unlike inverse rendering techniques that depend on specific scene configurations, our model can generalize to different lighting and viewing conditions as we match the effective BRDFs rather than rendered images.

Height field rendering. Rendering displacement-mapped surfaces closely relates to height field rendering. Self-shadowing on height fields can be computed in real-time by determining horizon angles for a set of azimuthal directions [92, 96]. Fast height field rendering with global illumination [76] approximates visibility and indirect radiance using low-order SH basis functions, which cannot capture high-frequency reflection and shadowing effects. In these methods, height fields are attached on a planar base surface. Our method allows perturbing displacements on a general surface representation such as triangular meshes.

4.3 Preliminaries

In this section, we provide preliminaries of displacement mapping, normal mapping, and effective BRDFs. Table 4.2 summarizes all symbols commonly used in this chapter.

Displacement mapping. Displacement maps provide an efficient way to describe detailed micro-geometries. Mathematically, a *displacement map* is a function $h : [0,1]^2 \to \mathbb{R}$ that specifies the distance which individual surface points are shifted along the normal directions. To be precise, given a texture-mapped base surface and a displacement map, the resulting geometry is obtained by moving each surface point with texture coordinate (u,v) along its normal direction for a distance of $h(u,v)$. Further, a *surface patch* $\mathcal{P} \subseteq [0,1]^2$ denotes a small and locally flat region of the base surface. For notational simplicity, the patch area is normalized using some filter kernel $k_{\mathcal{P}}$[1] (i.e., $\int_{\mathcal{P}} k_{\mathcal{P}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p} = 1$). For each $\boldsymbol{p} \in \mathcal{P}$, we denote its final position (perturbed by displacement mapping) and micro-normal as $\boldsymbol{x}_m(\boldsymbol{p})$ and $\boldsymbol{\omega}_m(\boldsymbol{p})$, respectively. Lastly, the micro-geometry given by a base patch $\mathcal{P}$ and a displacement map can be expressed by a collection

[1] We use a box filter in this chapter.

Table 4.2. Definitions of commonly used symbols.

Symbol	Meaning	Def.
$\mathcal{P}$	Surface patch	Section 4.3
$\mathcal{G}(\mathcal{P})$	Micro-geometry defined by a surface patch (subset) $\mathcal{P}$ of a displacement map	Section 4.3
f	Isotropic base BRDF	Section 4.3
$f^{\text{eff}}(\mathcal{G}, f)$	Effective BRDF determined by micro-geometry $\mathcal{G}$ and base BRDF f	Section 4.3, Equation (4.6)
$\mathcal{G}_{\text{orig}}$	Micro-geometry defined by the original high-resolution displacement map	Section 4.4
f_{orig}	Original input base BRDF	Section 4.4
$\mathcal{G}_{\text{low}}$	Micro-geometry defined by the low-resolution displacement map	Section 4.4
f_{low}	Prefiltered spatially varying base BRDF	Section 4.4
f'_{low}	Spatially varying multi-lobe BRDF	Section 4.4.2, Equation (4.11)
$f^{\text{eff}}_{\text{ir}}$	Effective BRDF with interreflections	Section 4.5.1, Equation (4.17)
R, R_{ir}	6D scaling function without/with interreflections	Section 4.4.3, Equation (4.12) Section 4.5.2, Equation (4.20)
T, T_{ir}	2D spatial scaling function without/with interreflections	Section 4.4.3, Equation (4.14) Section 4.6, Equation (4.21)
S, S_{ir}	4D angular scaling function without/with interreflections	Section 4.4.3, Equation (4.14) Section 4.6, Equation (4.21)

of micro-positions and micro-normals: $\mathcal{G}(\mathcal{P}) = \{(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_m(\boldsymbol{p})) \,|\, \boldsymbol{p} \in \mathcal{P}\}$.

Meso-geometry. For a surface patch $\mathcal{P}$, we define a local tangent frame under which the macro-normal $\boldsymbol{\omega}_g$ of the patch equals to $(0,0,1)$. Then, the meso-scale geometry of $\mathcal{G}(\mathcal{P})$ can be described with the *average slope*:

$$\tilde{\boldsymbol{s}} = \int_{\mathcal{P}} \boldsymbol{s}_m(\boldsymbol{p}) k_{\mathcal{P}}(\boldsymbol{p}) \, \mathrm{d}\boldsymbol{p}, \tag{4.1}$$

where $\boldsymbol{s}_m(\boldsymbol{p}) = (-x_m/z_m,\ -y_m/z_m)$ is the micro-slope with x_m, y_m, and z_m given by the micro-normal at $\boldsymbol{p}$ (i.e., $(x_m, y_m, z_m) = \boldsymbol{\omega}_m(\boldsymbol{p})$) under the local tangent frame.

Normal mapping. Provided a displacement map, we can extract the *normal distribution*

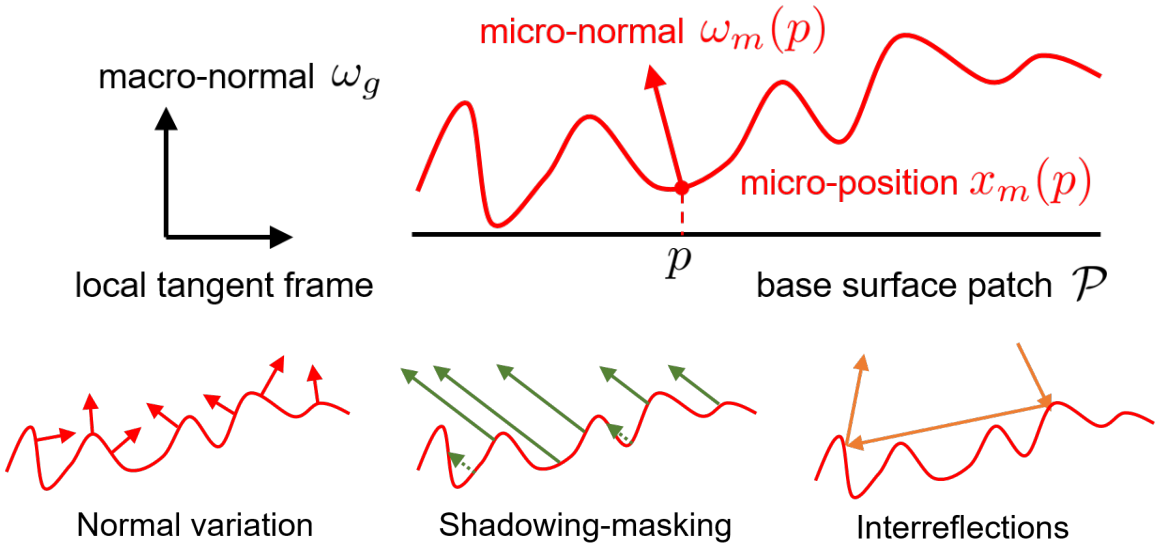


Figure 4.4. Illustrations of micro-geometry and related illumination effects. They are shown in 2D for simplicity, but fully developed in 3D.

function (NDF) [17] for any given surface patch $\mathcal{P}$ via

$$D(\boldsymbol{\omega}) = \int_{\mathcal{P}} \frac{\delta(\boldsymbol{\omega} - \boldsymbol{\omega}_m(\boldsymbol{p}))}{\langle \boldsymbol{\omega}_m(\boldsymbol{p}), \boldsymbol{\omega}_g \rangle} k_{\mathcal{P}}(\boldsymbol{p}) \, \mathrm{d}\boldsymbol{p}, \tag{4.2}$$

where $\langle \cdot, \cdot \rangle$ represents the dot product clamped to 0, and δ denotes the Dirac delta function. A physically valid NDF must satisfy

$$\int_{\mathcal{H}^2} D(\boldsymbol{\omega}) \langle \boldsymbol{\omega}, \boldsymbol{\omega}_g \rangle \, \mathrm{d}\boldsymbol{\omega} = \int_{\mathcal{P}} k_{\mathcal{P}}(\boldsymbol{p}) \, \mathrm{d}\boldsymbol{p} = 1, \tag{4.3}$$

where $\mathcal{H}^2$ denotes the unit hemispherical domain. It means that the projected area of the micro-geometry onto the macro-normal $\boldsymbol{\omega}_g$ should be equal to the area of the base surface patch.

An NDF D can be approximated by a mixture of von Mises-Fisher (vMF) lobes [25],

$$D(\boldsymbol{\omega}) \approx \sum_{i=1}^{m} \alpha_i \gamma(\boldsymbol{\omega}; \kappa_i, \boldsymbol{\mu}_i), \tag{4.4}$$

where α_i represents the lobe's amplitude. Each vMF lobe with bandwidth κ_i and the center direc-

tion $\boldsymbol{\mu}_i$ is defined as $\gamma(\boldsymbol{\omega};\kappa_i,\boldsymbol{\mu}_i)=\frac{\kappa_i}{4\pi\sinh\kappa_i}\exp(\kappa_i(\boldsymbol{\omega}\cdot\boldsymbol{\mu}_i))$. This vMF-based parameterization allows efficient description of spatially varying NDFs (SVNDFs).

From an NDF D and an isotropic base BRDF f, the composite multi-lobe BRDF can be formulated as [25]

$$\rho(\boldsymbol{\omega}_i,\boldsymbol{\omega}_o;D,f)=\frac{1}{\cos\theta_i}\int_{\mathcal{H}^2} f(R_{\boldsymbol{\omega}}(\boldsymbol{\omega}_i),R_{\boldsymbol{\omega}}(\boldsymbol{\omega}_o))\langle\boldsymbol{\omega},\boldsymbol{\omega}_i\rangle D(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}. \tag{4.5}$$

Note that $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ are in the macro-scale tangent frame defined by the macro-normal $\boldsymbol{\omega}_g$. We need a rotation function $R_{\boldsymbol{\omega}}(\cdot)$ to transform $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ to the local frame defined by the micro-normal $\boldsymbol{\omega}$. The multi-lobe BRDF keeps the orientations of the micro-normals, but neglects both shadowing-masking and interreflections caused by the micro-geometry.

Effective BRDFs. To include shadowing-masking effects, the effective BRDF f^{eff} is used to describe the overall reflectance of a patch $\mathcal{P}$ [107]. It depends on the micro-geometry $\mathcal{G}(\mathcal{P})$ and the micro-BRDF f. The effective BRDF can be viewed as the average of the cosine-weighted and shadowed micro-BRDFs weighted by the visible projected area along the viewing direction $\boldsymbol{\omega}_o$:

$$\begin{aligned} f^{\mathrm{eff}}(\boldsymbol{\omega}_i,\boldsymbol{\omega}_o;\mathcal{G},f)=\frac{1}{A_{\mathcal{G}}(\boldsymbol{\omega}_o)}\int_{\mathcal{P}} & f(\boldsymbol{x}_m(\boldsymbol{p}),\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\langle\boldsymbol{\omega}_m(\boldsymbol{p}),\boldsymbol{\omega}_i\rangle \\ & V(\boldsymbol{x}_m(\boldsymbol{p}),\boldsymbol{\omega}_i)A_{\mathcal{G}}(\boldsymbol{p},\boldsymbol{\omega}_o)k_{\mathcal{P}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p}, \end{aligned} \tag{4.6}$$

where $V(\boldsymbol{x},\boldsymbol{\omega})$ denotes the binary visibility function (indicating whether a ray starting from point $\boldsymbol{x}$ along direction $\boldsymbol{\omega}$ is occluded), and $A_{\mathcal{G}}(\boldsymbol{p},\boldsymbol{\omega}_o)$ is the visible projected area along $\boldsymbol{\omega}_o$ given by

$$A_{\mathcal{G}}(\boldsymbol{p},\boldsymbol{\omega}_o)=\frac{\langle\boldsymbol{\omega}_o,\boldsymbol{\omega}_m(\boldsymbol{p})\rangle}{\langle\boldsymbol{\omega}_g,\boldsymbol{\omega}_m(\boldsymbol{p})\rangle}V(\boldsymbol{x}_m(\boldsymbol{p}),\boldsymbol{\omega}_o). \tag{4.7}$$

The total visible projected area of $\mathcal{G}(\mathcal{P})$ is the normalization factor of the weighted average [17,

28],

$$A_{\mathcal{G}}(\boldsymbol{\omega}_o) = \int_{\mathcal{P}} \frac{\langle \boldsymbol{\omega}_o, \boldsymbol{\omega}_m(\boldsymbol{p}) \rangle}{\langle \boldsymbol{\omega}_g, \boldsymbol{\omega}_m(\boldsymbol{p}) \rangle} V(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_o) k_{\mathcal{P}}(\boldsymbol{p}) \, \mathrm{d}\boldsymbol{p} = \frac{\langle \boldsymbol{\omega}_o, \boldsymbol{\omega}_n \rangle}{\langle \boldsymbol{\omega}_g, \boldsymbol{\omega}_n \rangle}, \tag{4.8}$$

where $\boldsymbol{\omega}_n$ denotes the meso-scale normal direction.

The effective BRDF (Equation (4.6)) has the cosine term and the visibilities baked in and captures the shadowing-masking caused by micro-scale self-occlusions. We will generalize the formulation in Section 4.5.1 to capture interreflections.

4.4 Prefiltering Reflectance Parameters

Our method takes as input a high-resolution displacement map h_{orig} and an isotropic base micro-BRDF f_{orig} (which could be spatially varying, e.g., Figure 4.14). We denote $\mathcal{G}_{\text{orig}}(\mathcal{P})$ as the micro-geometry defined by the original displacement map on a surface patch $\mathcal{P}$. Then, we prefilter the input model and obtain a lower-resolution displacement map h_{low} (with $\mathcal{G}_{\text{low}}(\mathcal{P})$ representing its micro-geometry on $\mathcal{P}$) associated with a new spatially varying BRDF f_{low}. Our goal is to have the appearance of the prefiltered model closely resemble the input. Notice that, even if the input base BRDF f_{orig} is spatially invariant, the prefiltered reflectance f_{low} may need to have spatial variations to accurately reproduce the detailed appearance of the input model.

To this end, our technique starts with computing the downsampled displacement map h_{low} by minimizing differences of the meso-scale slopes (Section 4.4.1). Then, we seek a prefiltered SVBRDF f_{low} that preserves the original appearance for the downsampled displacement map. This, however, is nontrivial as f_{low} generally needs to be spatially varying. We introduce a novel two-step approach to compute f_{low}. First, for each base patch of the downsampled displacement map h_{low}, we compute its corresponding *patch NDF* from the original displacement map h_{orig}. Each patch NDF implies a spatially varying multi-lobe BRDF without considering shadowing-masking and interreflections. This step generates a spatially varying base BRDF f'_{low} as an initial solution (Section 4.4.2). Then, we scale f'_{low} by a 6D scaling function $R(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ to match

the effective BRDFs (Section 4.4.3). The final prefiltered SVBRDF $f_{\text{low}} = R \cdot f'_{\text{low}}$ is able to reproduce the original appearance. Our method is also illustrated in Figure 4.3.

Although the scaling function $R(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ is 6D, we will show in Section 4.6 that R can be factorized into a 2D spatial function of location $\boldsymbol{x}$ and a 4D angular function of directions $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$. Further, these functions are usually smooth in practice, and therefore can be computed efficiently as low-resolution tabulated functions, allowing our prefiltered models to be compactly represented.

4.4.1 Downsampling Displacement Maps

As stated in Section 4.3, displacement maps are defined mathematically as continuous 2D scalar functions. In practice, we represent displacement maps as piecewise linear functions using 2D textures. Specifically, the original high-resolution displacement map h_{orig} is defined at each vertex of a dense grid: $\{(u \cdot 2^{-l}, v \cdot 2^{-l})\}$ for $u, v = 0, \ldots, 2^l$. The prefiltered low-resolution displacement map h_{low} uses a coarse grid $\{(u \cdot 2^{-l'}, v \cdot 2^{-l'})\}$ for $u, v = 0, \ldots, 2^{l'}$ with some $l' < l$.

Obtaining h_{low} requires specifying the displacement values at the vertices of the coarse grid. For every patch $\mathcal{P}_{uv} = [u \cdot 2^{-l'}, (u+1) \cdot 2^{-l'}] \times [v \cdot 2^{-l'}, (v+1) \cdot 2^{-l'}]$ covering a grid cell of h_{low}, our goal is to find out the optimal heights at its four corners (i.e., grid points) such that the average slope $\tilde{\boldsymbol{s}}_{uv}$ of the bilinear patch closely matches the reference slope $\tilde{\boldsymbol{s}}^*_{uv}$ of the original micro-surface $\mathcal{G}_{\text{orig}}(\mathcal{P}_{uv})$. The reference average slope $\tilde{\boldsymbol{s}}^*_{uv}$ can be calculated using Equation (4.1). On the other hand, the average slope of the bilinear patch $\mathcal{P}_{uv}$ is

$$\tilde{\boldsymbol{s}}_{uv} = \left(\frac{h_{11} + h_{10} - h_{01} - h_{00}}{2}, \frac{h_{11} + h_{01} - h_{10} - h_{00}}{2} \right), \tag{4.9}$$

where $h_{00}, h_{10}, h_{01}, h_{11}$ denote the displacement values at the four cell corners. Please refer to Appendix B.1 for the derivation. To minimize the differences between the two average slopes robustly, we further introduce a regularization term for local smoothness, yielding the final

objective function:

$$L(h_{\text{low}}) = \sum_u \sum_v \|\tilde{\boldsymbol{s}}_{uv} - \tilde{\boldsymbol{s}}^*_{uv}\|^2 + w \left\| \Delta h_{\text{low}} \left(\frac{u}{2^{l'}}, \frac{v}{2^{l'}} \right) \right\|^2, \tag{4.10}$$

where Δ is the Laplacian operator and $w = 0.01$ is the weight of the regularization term. By minimizing this objective function using least-squares, the optimal solution gives the downsampled displacement map h_{low} and its corresponding micro-geometry $\mathcal{G}_{\text{low}}$.

4.4.2 Spatially Varying Multi-Lobe BRDF

For a cell from the coarse grid of the downsampled displacement map h_{low} and its corresponding patch $\mathcal{P}_{uv}$, the patch NDF $D_{uv}(\boldsymbol{\omega})$ defined in Equation (4.2) provides a compact approximation of the original micro-geometry $\mathcal{G}_{\text{orig}}(\mathcal{P}_{uv})$. Previous works such as LEADR [17] typically assume the NDFs to follow Beckmann distributions. Our method, in contrast, does not enforce any restriction on the NDFs and is therefore more general. We use the normal mapping technique [25] to fit an NDF with a mixture of vMF lobes (Equation 4.4), resulting in a spatially varying NDF (SVNDF) at a lower resolution. The SVNDF is technically 4D but can be described compactly using spatially varying vMF parameters (stored as 2D textures).

From these NDFs that approximate $\mathcal{G}_{\text{orig}}$ and the original base micro-BRDF f_{orig}, we use Equation (4.5) to formulate the initial multi-lobe BRDF for each patch $\mathcal{P}_{uv}$:

$$f'_{\text{low}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \rho(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; D_{uv}, f_{\text{orig}}). \tag{4.11}$$

Here the micro-position $\boldsymbol{x}$ is within the patch $\mathcal{P}_{uv}$.

4.4.3 Scaling Function

Although the multi-lobe SVBRDF f'_{low} is a good start for matching the input appearance, it is incomplete as it neglects the shadowing-masking and interreflection effects. To address this problem, we first consider the single-bounce case (i.e., direct illumination that only involves

shadowing and masking). We will discuss the multiple-bounce case that handles interreflections in Section 4.5.

To capture shadowing and masking, we multiply the initial multi-lobe SVBRDF f'_{low} with another scaling function R. Suppose R has a spatial resolution of M^2, we uniformly subdivide the base surface into M^2 patches. Let $\mathcal{P}_{\boldsymbol{x}}$ denote the patch containing $\boldsymbol{x}$. We define the scaling function $R(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ as the ratio[2] between the original effective BRDF (Equation 4.6) and the prefiltered effective BRDF over $\mathcal{P}_{\boldsymbol{x}}$:

$$\boxed{R(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{f^{\text{eff}}\left(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \mathcal{G}_{\text{orig}}(\mathcal{P}_{\boldsymbol{x}}), f_{\text{orig}}\right)}{f^{\text{eff}}\left(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \mathcal{G}_{\text{low}}(\mathcal{P}_{\boldsymbol{x}}), f'_{\text{low}}\right)}}, \tag{4.12}$$

The final prefiltered base SVBRDF can then be expressed as

$$f_{\text{low}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = R(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) f'_{\text{low}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o). \tag{4.13}$$

It is easy to verify that the final effective BRDF $f^{\text{eff}}(\mathcal{G}_{\text{low}}, f_{\text{low}})$ (with $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ omitted for notational simplicity) over $\mathcal{P}_{\boldsymbol{x}}$ equals the original effective BRDF, since the single-bounce effective BRDF is linear in the scaling factors, i.e., $f^{\text{eff}}(\mathcal{G}_{\text{low}}, R \cdot f'_{\text{low}}) = R \cdot f^{\text{eff}}(\mathcal{G}_{\text{low}}, f'_{\text{low}})$. The identical effective BRDFs indicate a good match of aggregate reflectance between the original and the prefiltered models.

Due to the high dimensionality of the scaling function R, it is challenging to obtain and store. We will show in Section 4.6 that R can be accurately approximated as the product of a function $T(\boldsymbol{x})$ of location and a function $S(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ of directions:

$$\boxed{R(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \approx T(\boldsymbol{x}) \cdot S(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)}. \tag{4.14}$$

We first generalize our scaling function to interreflections in Section 4.5. Then, in Section 4.6, we develop the factorization above and show that the factorized scaling functions T and S can be

[2] In practice, we need to add a small ε to avoid division by zero.

computed directly without fully computing the 6D function R.

4.5 Interreflections

We now describe how to capture interreflection effects using a scaling function similar to Equation (4.12). We first generalize the effective BRDF formulation (Equation (4.6)) to measure the average surface reflectance with interreflections (Section 4.5.1). Then, we provide an approximate solution to the scaling function with interreflections handled (Section 4.5.2).

4.5.1 Effective BRDF with Interreflections

The effective BRDF formulation depicted in Section 4.3 models only direct (i.e., single-bounce) illumination of the micro-geometry. This leads to significant energy loss due to the neglect of interreflections within the micro-geometry. To address this problem, we adapt Veach's path integral formulation [99] to express effective BRDFs with interreflections.

A ray that bounces within the micro-geometry $\mathcal{G} = \{(\boldsymbol{x}_m, \boldsymbol{\omega}_m)\}$ can be described by a light transport path $\bar{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ as well as the viewing direction $\boldsymbol{\omega}_o$ and the lighting direction $\boldsymbol{\omega}_i$. Notice that, unlike the traditional formulation where the endpoints of a path respectively lay on the light source and the sensor, all the path vertices $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are located on the micro-surface in our case. The path contribution $h(\bar{x})$ is given by the product of BRDF terms and cosine terms, followed by a visibility term at the last path vertex $\boldsymbol{x}_k$ along the lighting direction $\boldsymbol{\omega}_i$:

$$h(\bar{x}) = V(\boldsymbol{x}_k, \boldsymbol{\omega}_i) \prod_{j=1}^{k} f(\boldsymbol{x}_j, \boldsymbol{\omega}_{j\to j+1}, \boldsymbol{\omega}_{j\to j-1}) \langle \boldsymbol{n}(\boldsymbol{x}_j), \boldsymbol{\omega}_{j\to j+1} \rangle, \tag{4.15}$$

where $\boldsymbol{\omega}_{i\to j} := \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}$ indicates the normalized direction from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ (with $\boldsymbol{\omega}_{1\to 0} := \boldsymbol{\omega}_o$ and $\boldsymbol{\omega}_{k\to k+1} := \boldsymbol{\omega}_i$), and $\boldsymbol{n}(\boldsymbol{x}_j)$ denotes the micro-normal at $\boldsymbol{x}_j$.

We define the path space $\Omega(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ as a collection of light transport paths $\bar{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ with $\boldsymbol{x}_1 = \boldsymbol{x}$, and the viewing and illumination directions given by $\boldsymbol{\omega}_o$ and $\boldsymbol{\omega}_i$,

respectively. At a given position $\boldsymbol{x}_m(\boldsymbol{p}) \in \mathcal{G}$, the reflectance including interreflections aggregates contributions from all the valid paths:

$$r(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \int_{\Omega(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)} h(\bar{x}) \, \mathrm{d}\mu(\bar{x}), \tag{4.16}$$

where $\mathrm{d}\mu(\bar{x})$ denotes the path throughput measure, which is a product of solid angle measures on the path directions.

According to Equations (4.6) and (4.16), we express the multi-bounce effective BRDF as the average path contribution over $\mathcal{G}$ weighted by the visible projected area along $\boldsymbol{\omega}_o$:

$$f_{\mathrm{ir}}^{\mathrm{eff}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \mathcal{G}, f) = \frac{1}{A_{\mathcal{G}}(\boldsymbol{\omega}_o)} \int_{\mathcal{P}} r(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) A_{\mathcal{G}}(\boldsymbol{p}, \boldsymbol{\omega}_o) k_{\mathcal{P}}(\boldsymbol{p}) \, \mathrm{d}\boldsymbol{p}. \tag{4.17}$$

Under this formulation, the single-bounce effective BRDF (Equation (4.6)) becomes a special case where each path $\bar{x}$ is restricted to contain only one vertex.

4.5.2 Computing the Scaling Function

Following a similar idea as Section 4.4.3, we scale the initial multi-lobe SVBRDF f'_{low} by a scaling function R_{ir}:

$$f_{\mathrm{low}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = R_{\mathrm{ir}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) f'_{\mathrm{low}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o), \tag{4.18}$$

so that the resulting multi-bounce effective BRDF (Equation (4.17)) of the prefiltered model matches that of the input. Namely, $f_{\mathrm{ir}}^{\mathrm{eff}}(\mathcal{G}_{\mathrm{low}}, f_{\mathrm{low}}) = f_{\mathrm{ir}}^{\mathrm{eff}}(\mathcal{G}_{\mathrm{orig}}, f_{\mathrm{orig}})$. Since $f_{\mathrm{ir}}^{\mathrm{eff}}$ is nonlinear in R_{ir} due to multiple scattering (i.e., $f_{\mathrm{ir}}^{\mathrm{eff}}(\mathcal{G}_{\mathrm{low}}, R_{\mathrm{ir}} \cdot f'_{\mathrm{low}}) \neq R_{\mathrm{ir}} \cdot f_{\mathrm{ir}}^{\mathrm{eff}}(\mathcal{G}_{\mathrm{low}}, f'_{\mathrm{low}})$ in general), R_{ir} cannot be obtained using Equation (4.12). Although one can optimize R_{ir} iteratively using inverse rendering methods [50, 121], the optimization would be very expensive, if not impractical, due to the large number of unknowns involved.

Instead, we seek R_{ir} without performing global optimizations. To this end, we make a

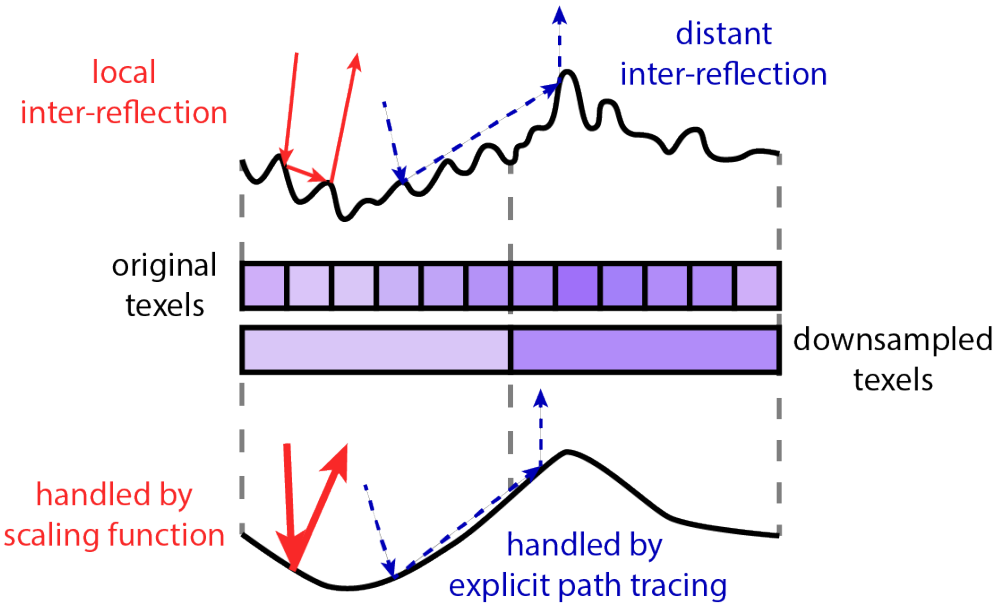


Figure 4.5. Local and global interreflections. Our method tries to match local interreflections using a scaling function, while the path tracer handles distant (global) interreflections.

simplifying assumption: the high-order components in $f_{\text{ir}}^{\text{eff}}(\mathcal{G}_{\text{low}}, f_{\text{low}})$ are negligible. In other words,

$$f_{\text{ir}}^{\text{eff}}(\mathcal{G}_{\text{low}}, f_{\text{low}}) \approx f^{\text{eff}}(\mathcal{G}_{\text{low}}, f_{\text{low}}). \tag{4.19}$$

This is because the high-order component mainly consists of *distant* interreflections (i.e., those across multiple surface patches at the reduced resolution) that are supposed to be handled with explicit path tracing (see the blue arrows in Figure 4.5). The *local* interreflection, on the other hand, is usually weakened due to the downsampling of an object's micro-geometry, which we compensate using the scaling function R_{ir} (see the red arrows in Figure 4.5).

Under this formulation, our goal is to match $f_{\text{ir}}^{\text{eff}}(\mathcal{G}_{\text{orig}}, f_{\text{orig}})$ and $R_{\text{ir}} \cdot f^{\text{eff}}(\mathcal{G}_{\text{low}}, f'_{\text{low}})$, yielding the scaling function:

$$R_{\text{ir}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{f_{\text{ir}}^{\text{eff}}\left(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o;\ \mathcal{G}_{\text{orig}}(\mathcal{P}_{\boldsymbol{x}}), f_{\text{orig}}\right)}{f^{\text{eff}}\left(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o;\ \mathcal{G}_{\text{low}}(\mathcal{P}_{\boldsymbol{x}}), f'_{\text{low}}\right)}. \tag{4.20}$$

In Section 4.8, we show that the simplification usually has minimal affect on accuracy (Fig-

ures 4.17, 4.18).

4.6 Properties of the scaling function

Upon establishing the scaling function R_{ir} (Equation (4.20)), our problem of prefiltering displacement-mapped surfaces boils down to efficiently representing and computing this function,[3] which, unfortunately, is nontrivial as R_{ir} is 6D and generally too expensive to compute and store in brute-force ways.

We tackle this challenge by factorizing the 6D scaling function $R_{\text{ir}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ into a single spatial scaling function $T_{\text{ir}}(\boldsymbol{x})$ and a single angular scaling function $S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$. In practice, both T_{ir} and S_{ir} are smooth (i.e., low-frequency). We exploit their smoothness to achieve efficient factorization using only a sparse sampling of R_{ir} (without explicitly precomputing the full 6D function). In the rest of this section, we provide more details on our factorization method.

Rank-1 factorization. Our experiments indicate that the scaling functions are typically low-rank (Figure 4.6). To demonstrate this property, we compute the full 6D scaling functions for two example models: "silk" with only direct illumination; and "twill" with global illumination (details for these models are described in Section 4.8.2). In both cases, the scaling matrices R_{ir} are tabulated with a resolution $16^2 \times 15^2 \times 15^2$ (16^2 for $\boldsymbol{x}$, 15^2 for $\boldsymbol{\omega}_i$, 15^2 for $\boldsymbol{\omega}_o$).[4] After excluding the directions near grazing angles ($\theta > 75^\circ$),[5] we reshape R_{ir} into a 256×11^4 matrix, which is then decomposed using singular value decomposition (SVD). For both examples, the greatest singular value carries over 95% of the total energy (see Figure 4.6), suggesting that the scaling function R_{ir} can indeed be accurately approximated by a rank-1 factorization:

$$R_{\text{ir}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \approx T_{\text{ir}}(\boldsymbol{x}) \cdot S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o). \tag{4.21}$$

[3] We use the notation R_{ir} in the multiple-bounce case since it is more general. The properties also hold for R in the single-bounce case.

[4] This resolution is adequate due to the smoothness of R_{ir}, which will be discussed later in this section.

[5] Numerical evaluation of the scaling functions is unstable near grazing angles.

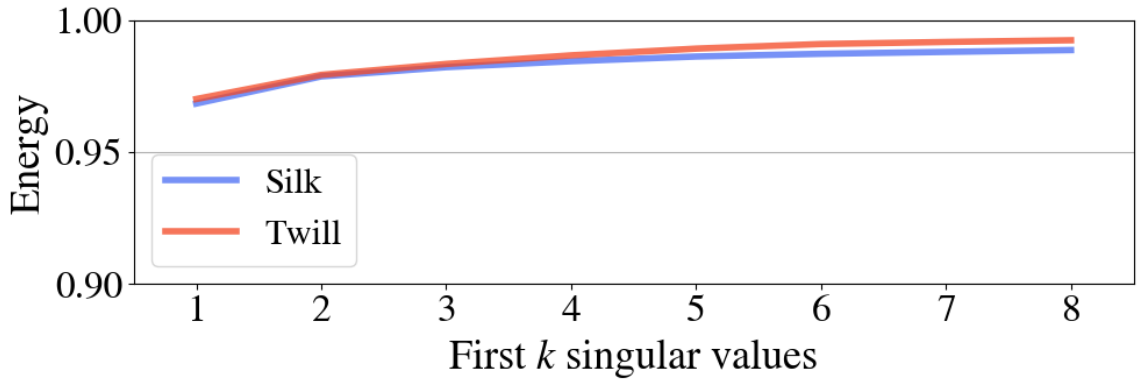


Figure 4.6. Graph showing the fractions of energy carried by the top eight singular values of the scaling matrix.

Angular smoothness. We now seek the optimal resolution for the scaling function $R_{\text{ir}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ that provides a good balance between accuracy and data size, starting with the angular resolution N. Specifically, we use the concentric mapping [87] to warp a unit hemisphere onto a unit square. Then, both $\boldsymbol{\omega}_i$ are $\boldsymbol{\omega}_o$ are sampled from a regular 2D grid with resolution N^2.

Because an effective BRDF captures the aggregate reflectance over a micro-surface, it generally has low angular frequency. Thus, the scaling function, as the ratio of two effective BRDFs, is also expected to be low-frequency angularly. Figure 4.7 shows 2D slices of the scaling functions (with $\boldsymbol{\omega}_i$ changing and the other parameters fixed) for different angular resolutions $N = 5, 15, 25$ and 45 (as references). The results indicate that $N = 15$ provides good accuracy, which is further supported by renderings in Figure 4.8. Therefore, we choose the resolution of $N = 15$ for all main results in this chapter.

Spatial smoothness. To determine the spatial resolution M, we perform a similar experiment by comparing renderings using resolutions of $M = 1, 4, 16$. As shown in Figure 4.9, the results indicate that increasing the spatial resolution has diminishing benefits. This is because the scaling function varies in a very smooth fashion spatially. We thus use $M = 4$ in most of our results (unless we state explicitly). In certain cases, even using $M = 1$ (that causes the scaling function T_{ir} to be constant-valued) can offer sufficient accuracy.

Efficient factorization. Although computing the full 6D scaling function R_{ir} and decom-

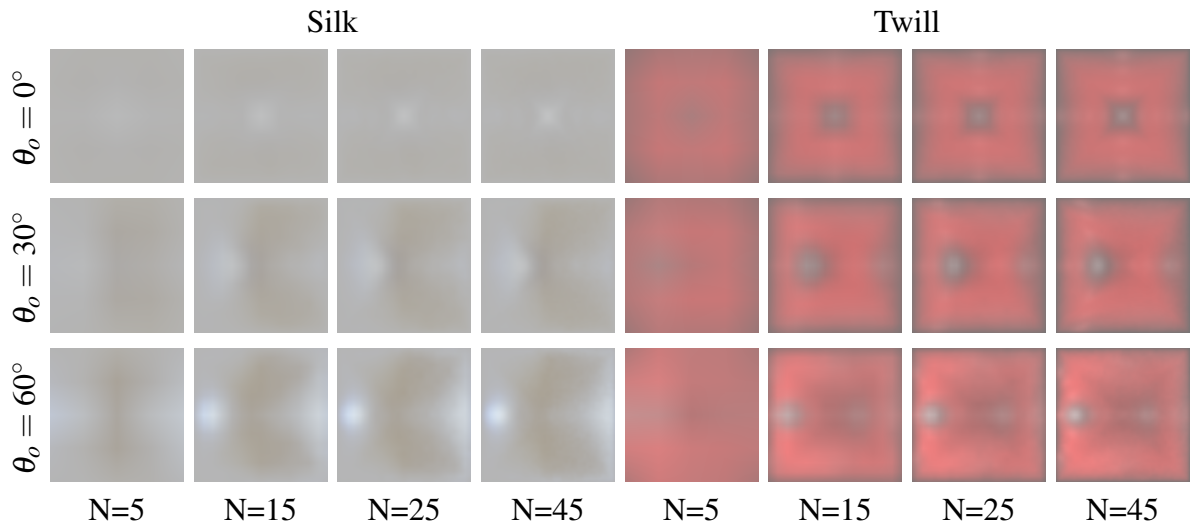


Figure 4.7. Scaling function slices (with only $\boldsymbol{\omega}_i$ changing and the other parameters fixed) for different angular resolutions. Using $N = 15$ provides adequate accuracy for describing the scaling functions.

posing it using SVD provides optimal factorization results, doing so is very costly since both steps require massive computation. We introduce a simple and efficient factorization method that constructs $T_{\text{ir}}(\boldsymbol{x})$ and $S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ from sparse samples of R_{ir}.

According to Equation (4.21), $T_{\text{ir}}(\boldsymbol{x})$ can be computed by averaging the 6D scaling function R_{ir} over the angular domain:

$$T_{\text{ir}}(\boldsymbol{x}) = \frac{1}{C} \int_{\mathcal{H}^2} \int_{\mathcal{H}^2} R_{\text{ir}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \, \mathrm{d}\boldsymbol{\omega}_i \, \mathrm{d}\boldsymbol{\omega}_o, \tag{4.22}$$

where

$$C = \int_{\mathcal{P}_{\text{all}}} \int_{\mathcal{H}^2} \int_{\mathcal{H}^2} R_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p}) \, \mathrm{d}\boldsymbol{\omega}_i \, \mathrm{d}\boldsymbol{\omega}_o \, \mathrm{d}\boldsymbol{p}, \tag{4.23}$$

is a normalization factor, and $\mathcal{P}_{\text{all}} = [0,1]^2$ covers the whole displacement map. To obtain T_{ir} as a tabulated function, we estimate its value at each bin using Monte Carlo integration based on a sparse (joint) sampling of $\boldsymbol{x}$ (among all locations within the bin), $\boldsymbol{\omega}_i$, and $\boldsymbol{\omega}_o$. We provide more implementation details in Section 4.7.

On the other hand, the angular scaling function $S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ can be obtained by averaging

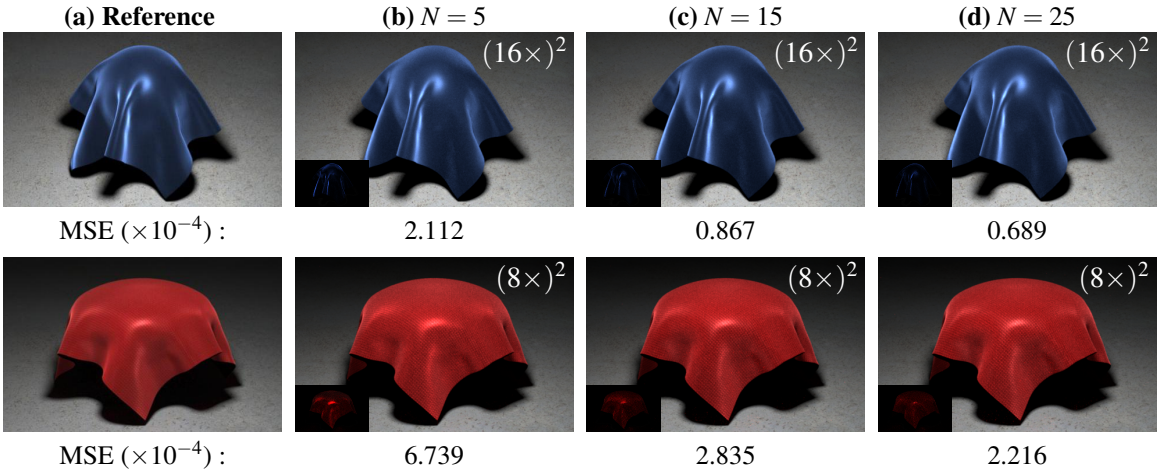


Figure 4.8. Using varying angular resolutions. Increasing angular resolution improves rendering accuracy, but at the expense of larger computational and storage overhead (with a fixed spatial resolution of $M = 1$). We use a resolution of $N = 15$, which shows the best tradeoff between accuracy and storage/performance. The insets (and those in the next figures 4.9, 4.10(e, f)) show $10\times$ squared errors compared to the reference images.

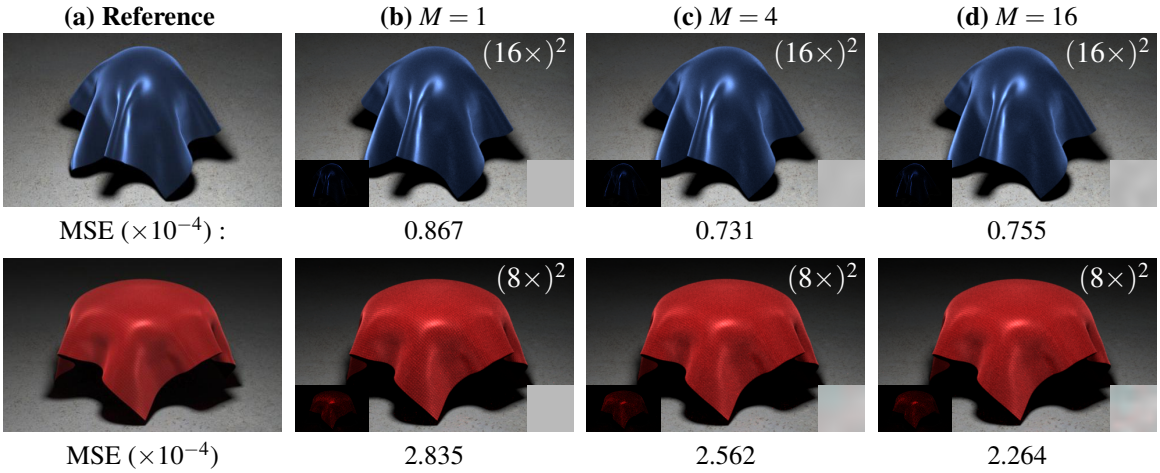


Figure 4.9. Using varying spatial resolutions. Increasing spatial resolutions results in more accurate rendering results, but the improvement is diminishing after $M = 4$ (errors may become slightly larger because of Monte Carlo noise). We find that $M = 4$ is adequate in most cases, and even using $M = 1$ yields high-quality results. Bottom-right insets visualize the 2D spatial scaling functions for different resolutions.

R_{ir} spatially:

$$S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \int_{\mathcal{P}_{\text{all}}} R_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p}) \, \mathrm{d}\boldsymbol{p}. \tag{4.24}$$

As a tabulated function, S_{ir} can be computed in a similar way as T_{ir} using Monte Carlo integration.

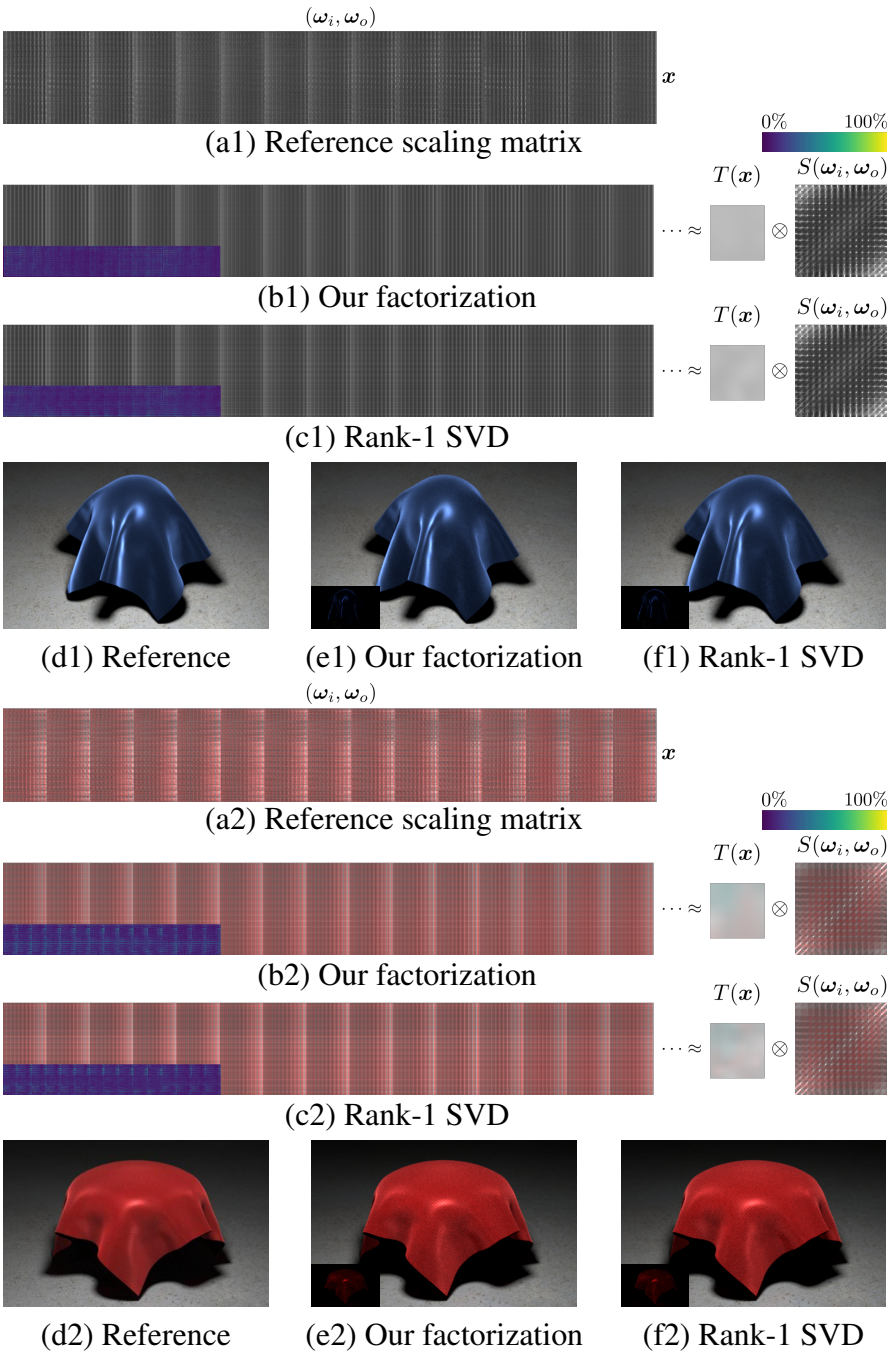


Figure 4.10. Factorization comparison. We compare our factorization method to rank-1 SVD in terms of scaling function reconstruction (a–c) and rendered images (d–f). We show 6D scaling functions (they are reorganized as scaling matrices) in (a). Even without explicitly computing the full 6D functions, the reconstruction error of our factorization (b) is similar to the error given by rank-1 SVD (c). Both errors are small because of the low-rank nature of the scaling matrices. The insets in (b, c) show the relative error maps of the reconstructed matrices compared to the reference matrices. The colored error bar is shown on the top-right. In terms of visual quality, using the factorized spatial scaling function T and angular scaling function S computed by our method reproduces the reference appearance accurately.

It is easy to verify that our definitions of T_{ir} and S_{ir} in Equations (4.22) and (4.24)) are consistent with Equation (4.21). Please see Appendix B.2 for more details.

Figure 4.10 compares factorization results obtained using our method and SVD (that uses full R_{ir}). Our method offers similar reconstruction errors as SVD does, which is demonstrated in Figure 4.10(b, c). Further, the corresponding renderings shown in Figure 4.10(d–f) indicate that the loss in visual quality caused by our method is negligible.

4.7 Implementation

We now provide details of our implementation. First, we provide details on how our technique works at individual scales (Section 4.7.1). Then, we describe how mipmaps of our prefiltered models can be created, enabling anti-aliased level-of-detail (LoD) rendering of displacement-mapped models (Section 4.7.2).

4.7.1 Prefiltering at a Single Scale

Provided a high-resolution displacement map h_{orig} and a downsampling scale, we first optimize for the downsampled displacement map using least-squares (Section 4.4.1) and compute the initial spatially varying multi-lobe BRDF f'_{low} (Section 4.4.2). As in prior work [25], we use six vMF lobes per patch NDF. Then, we seek the scaling function R_{ir} so that the final SVBRDF $f_{\text{low}} = R_{\text{ir}} \cdot f'_{\text{low}}$ preserves the appearance of the input model (Sections 4.4.3 and 4.5.2). This is achieved using our efficient factorization of R_{ir} (Section 4.6).

To estimate the value of T_{ir} at each tabulation bin, we use 5000 pairs of $(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ with both directions independently sampled from a cosine-weighted distribution. For evaluating the scaling function R_{ir} via Equation (4.20), we trace 2500 paths to estimate the effective BRDFs (Equation 4.17). Lastly, we normalize T_{ir} such that it averages to one. To estimate the scaling function S_{ir} of directions, we use 16 stratified samples for $\boldsymbol{p}$ and 2500 light transport paths for R_{ir}. We find these sampling rates are sufficient to reproduce accurate appearances.

We implement our method based on the Mitsuba physically based renderer [41]. Running

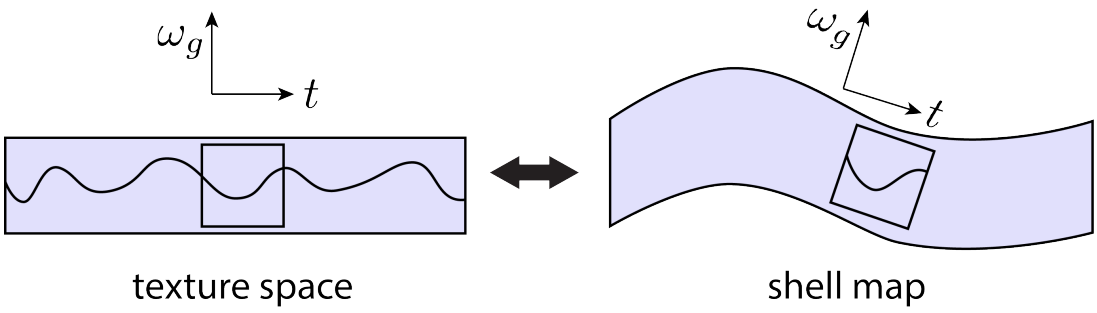


Figure 4.11. Shell map illustration. We deform a flat displacement map to a general shape using shell maps.

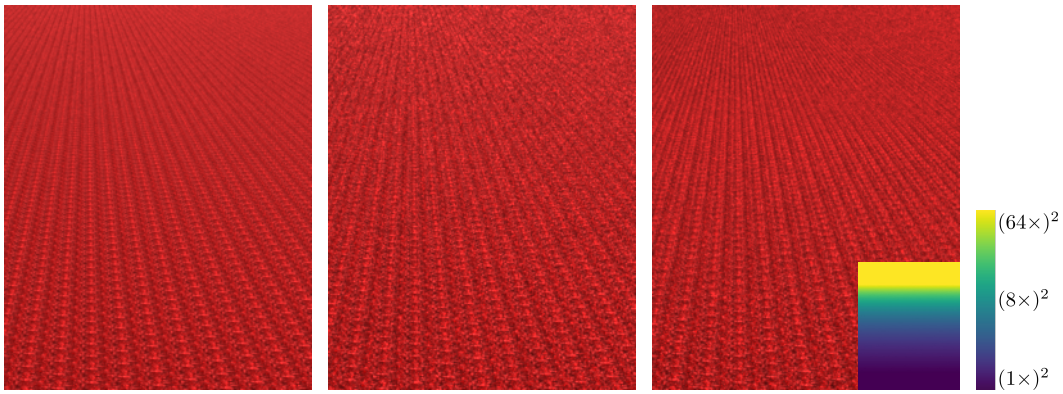
on a workstation equipped with a six-core Intel i7-5930K CPU, computing the factorized functions T_{ir} and S_{ir} takes 20–30 minutes for each input model.

Shell mapping. We use shell mapping [81] to deform a flat displacement map to a general shape (Figure 4.11), instead of explicitly displacing a base mesh. In fact, objects in our scenes are perturbed by extreme high-resolution displacement maps (resolution up to $(200K)^2$, tiled by $(1K)^2$ base displacement maps) to describe very small geometric details. Creating an explicit displacement mesh is not practical.

Importance sampling. Our final prefiltered SVBRDF f_{low} equals to the product of two functions R_{ir} and f'_{low}. Due to the smoothness of the scaling function R_{ir}, we simply follow f'_{low} when importance sampling f_{low}. That said, product importance sampling [13, 32] could be used to further improve the sampling quality.

4.7.2 Level of Detail

To enable anti-aliased rendering at multiple scales, we prefilter the input model at several downsampling scales. The resulting models effectively form a mipmap. At each mipmap level, we compute and store the lobe parameters as well as the scaling functions T_{ir} and S_{ir}, in addition to the downsampled displacement map. To balance computational and storage overhead and rendering performance, our mipmap uses a sparser set of levels compared to conventional texture mipmaps. In other words, the scales of two contiguous levels of our mipmaps usually differ



(a) Reference
Noise level:

(b) Original
0.0162

(c) LoD (linear)
0.0138

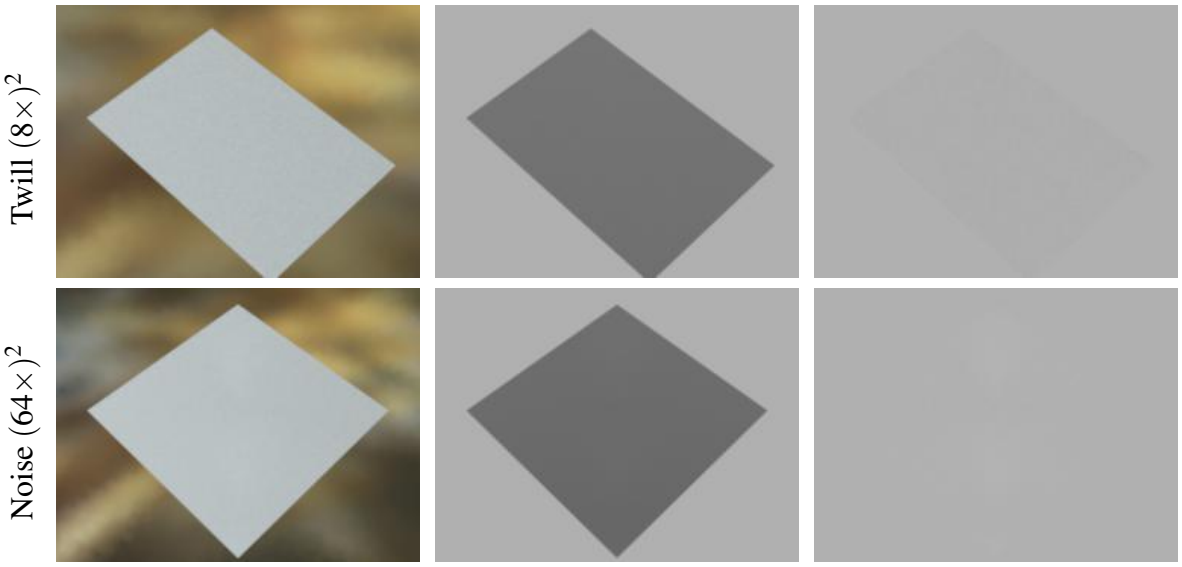
Figure 4.12. LoD rendering. The reference (a) is rendered using a high-resolution displacement map and 5K spp. The baseline result (b) is rendered using the same displacement map but $(10\times)$-lower spp. Our equal-time low-spp LoD rendering (c) closely matches the reference (a) at multiple scales, and suffers from less noise (measured using Liu et al. [64], lower is better) compared to the baseline (b). The mipmap levels are shown in inset.

by more than $2\times$. For example, the mipmap of the "silk" model includes $(16\times)^2$- and $(64\times)^2$-downsampled versions (besides the original model), and the mipmap of "twill" has $(8\times)^2$ and $(64\times)^2$ downsampled versions.

At render time, we use ray differentials [37] to compute each pixel's footprint on the original displacement map and determine the desired mipmap levels. To interpolate between two levels, we trace paths on two different models and linearly interpolate the path contributions. Please refer to Figures 4.1, 4.12 and the accompanying video in the original publication for results rendered with this method.

4.8 Results

We now show results generated using our method: Section 4.8.1 contains experimental evaluations and justifications; Section 4.8.2 demonstrates the effectiveness of our technique via a few examples. Finally, we discuss limitations of our methods and future research directions (Section 4.8.3).



(a) Environmental light (b) Constant light with R (c) Constant light with R_{ir}

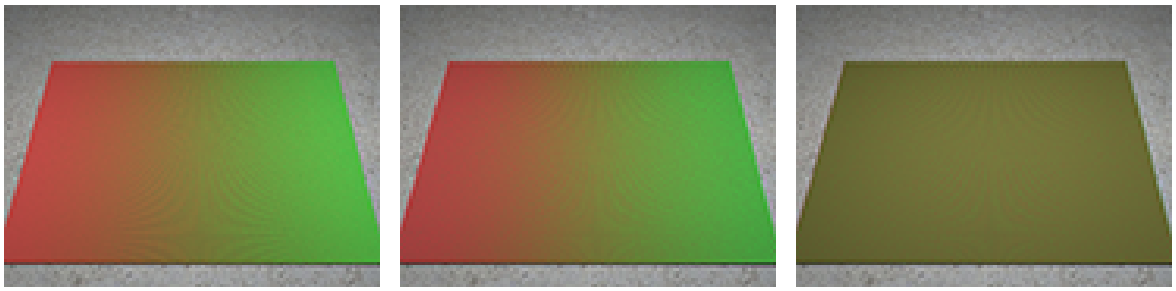
Figure 4.13. White furnace test. We prefilter displacement maps with a non-absorptive BRDF. Using the scaling function R handling only direct illumination (i.e., single bounce) leads to significant energy loss. Taking interreflections into account, the object rendered with the multi-bounce scaling function R_{ir} becomes almost invisible, showing negligible energy loss.

4.8.1 Evaluations and Justifications

Energy conservation. We demonstrate that our prefiltered models conserve energy via white furnace tests (Figure 4.13). Given a non-absorptive base BRDF and a constant environment light, our prefiltered models without interreflections (Figure 4.13(b)) lose a significant amount of energy, while those capturing interreflections (Figure 4.13(c)) preserve most of the energy. The effective albedos of our prefiltered models in Figure 4.13(c) are 0.988 (twill) and 0.999 (noise).

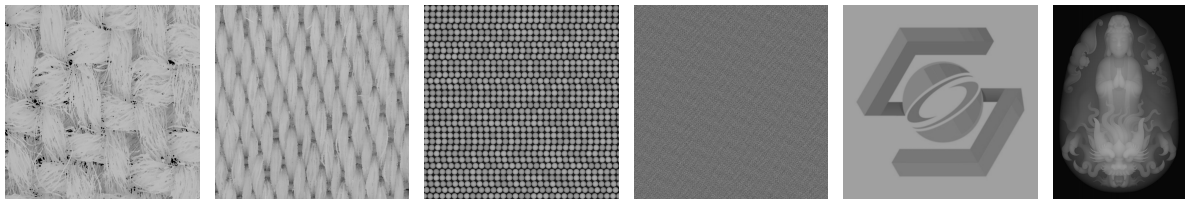
The ground truth effective BRDFs are energy conserving as they are determined by explicitly simulating light transport within the micro-surfaces. Theoretically, our prefiltered models as approximations of the ground truth effective BRDFs are not guaranteed to conserve energy. Limited energy loss is mainly due to neglecting distant interreflections. In practice, however, we do not observe any problem with energy conservation.

Reciprocity. Our effective BRDFs (Equation 4.17) are reciprocal when weighted by visible projected area (similar to the microflake reciprocity constraints proposed by Heitz et al. [29]):



(a) Reference (b) Ours ($R_{\mathrm{ir}} \cdot f'_{\mathrm{low}}$) (c) Multi-Lobe ($f'_{\mathrm{low}}$)

Figure 4.14. Correlated surface. We prefilter a two-color sawtooth (i.e., V-groove) surface. Thanks to the scaling function R_{ir}, our prefiltered model can effectively capture the normal-color correlation and accurately reproduce the appearance.



(a) Twill (b) Silk (c) Bump (d) Noise (e) Logo (f) Relief

Figure 4.15. Displacement maps used in our results.

$$f_{\mathrm{ir}}^{\mathrm{eff}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \mathcal{G}, f) A_{\mathcal{G}}(\boldsymbol{\omega}_o) = f_{\mathrm{ir}}^{\mathrm{eff}}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i; \mathcal{G}, f) A_{\mathcal{G}}(\boldsymbol{\omega}_i). \tag{4.25}$$

Since we aim to match the effective BRDFs given by the input models, our prefiltered models are approximately reciprocal.

Correlated surfaces. Our prefiltering method can handle surfaces with correlations. Figure 4.14 shows a synthetic two-color sawtooth example where the surface normals and colors are strongly correlated. The baseline multi-lobe BRDFs f'_{low} disregard this correlation and produce wrong appearance. Our result, on the other hand, captures this correlation properly and preserves the reference appearance accurately.

Table 4.3. Model Information. We list the detailed parameters used for prefiltering the input model at a single downsampling scale. We also provide the data size of our prefiltered models.

	Original model				Our prefiltered model					
Name	Reso.	Tiling	Total size (KB)	Base BRDF	Downsampling scale	Total size (KB)	h_{low} (KB)	f'_{low} (KB)	T/T_{ir} (KB)	S/S_{ir} (KB)
Twill	$(1K)^2$	100^2	3502	glossy	$(8\times)^2$	2377	65	1915	1	396
Silk	$(1K)^2$	200^2	2724	glossy	$(16\times)^2$	1085	17	478	1	589
Bump	$(1K)^2$	4^2	4015	diffuse	$(16\times)^2$	704	17	484	1	202
Noise	$(1K)^2$	100^2	3954	diffuse	$(64\times)^2$	239	2	34	1	202
Logo[6]	$(1K)^2$	$(1K)^2$	723	glossy	$(1K\times)^2$	589	~ 0	2	1	586
Relief	$(4K)^2$	1^2	33114	glossy	$(16\times)^2$	8367	238	7619	0 (M=1)	510

4.8.2 Main Results

We use two sets of scenes to demonstrate the effectiveness of our method. The scene objects and their detailed parameters (before and after prefiltering) are summarized in Table 4.3. We also present the precomputation time of our prefiltering method in Table 4.4. The first set contains two fabric examples (Figures 4.2, 4.16, and 4.17). The "twill" and "silk" models (Figure 4.15(a, b)) are extracted from detailed volume data [118]. The resulting surface micro-geometries are highly complex and can cause severe aliasing. The "plane" scene in Figure 4.2 contains a flat twill fabric. The "twill" scene in Figures 4.16 and 4.17 (top row) uses the same fabric but shell-mapped. The "silk" scene in Figures 4.16 and 4.17 (bottom row) has another shell-mapped silk fabric. Although we use isotropic glossy BRDFs in these scenes, the rendered images still show anisotropic glossy highlights because of the complex structured micro-geometries.

Another set of scenes contains objects with more diverse materials. In Figure 4.18 (top row), we show the "ball" scene that has a bumpy ball on a rough floor. The ball is represented by the "bump" model (Figure 4.15(c)), while the floor is represented by the "noise" model (Figure 4.15(d)). Both models use diffuse base BRDFs. In Figure 4.18 (middle row), we show

[6] We provide only parameters of the small-scale displacement map, since the large-scale one remains unchanged when downsampling.

Table 4.4. We list the precomputation time for each individual step of our prefiltering method. Time is measured in minutes.

Name	Total time	h_{low} and f'_{low}	T or T_{ir}	S or S_{ir}
Twill	22	1	7	14
Silk	22	1	7	14
Bump	24	1	3	20
Noise	22	1	4	17
Logo	36	1	12	23
Relief	26	1	0	25

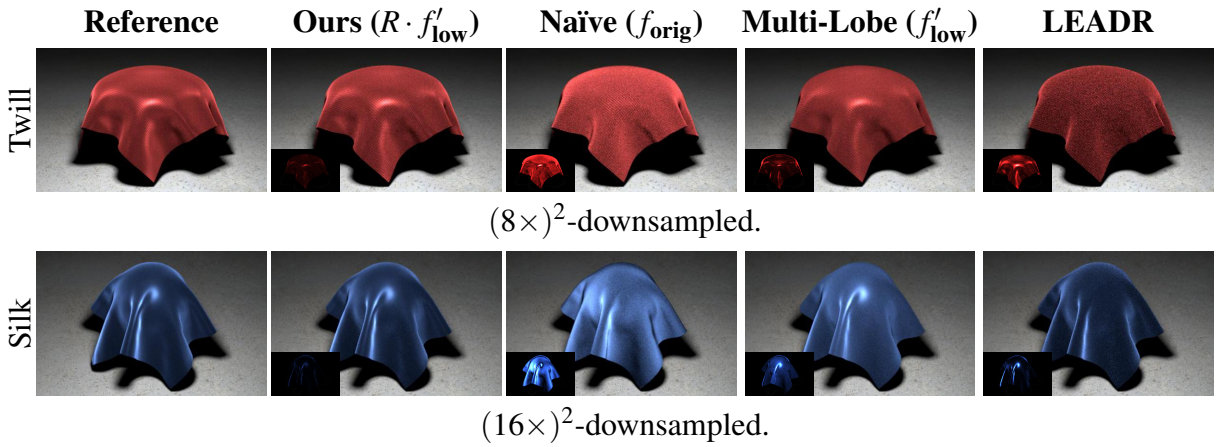


Figure 4.16. Direct illumination results. Our method accurately matches the reference even with significant downsampling, and is much more accurate than naive downsampling, multi-lobe BRDFs without taking shadowing-masking or interreflections into account, and LEADR. The insets (and those in the next figures 4.17, 4.18) show $10\times$ squared errors compared to the reference images.

the "logo" scene that is represented by a bi-scale displacement map (Figure 4.15(e)). The large-scale displacement map describes the overall shape, while the small-scale displacement map (which is a tiled version of the large-scale one) accounts for the detailed micro-geometry. In Figure 4.18 (bottom row), we show the "relief" scene that is represented by a composite displacement map (Figure 4.15(f), which is resampled to keep the image aspect ratio). It is constructed by adding procedural Perlin noise on a height field converted from a gray scale image. In this example, we use a spatial scaling function with resolution $M = 1$ since it leads to sufficient accuracy.

Table 4.5. Equal-time performance. Our prefiltered models not only reduce the storage size, but also bring benefits in rendering. Given equal time (around 60s for 50–100 spp), rendering using our models results in anti-aliased images, which have lower MSE than those rendered using the original models.

Scene	Twill	Silk	Ball	Logo	Relief
Original model's MSE ($\times 10^{-3}$)	1.095	1.269	2.055	2.072	2.946
Our model's MSE ($\times 10^{-3}$)	0.571	0.622	1.850	0.498	1.692

Direct illumination. Figure 4.16 contains rendered images generated through the intermediate steps of our method with the effective BRDFs capturing only direct illumination (i.e., neglecting micro-scale interreflections). Directly using original BRDFs f_{orig} with downsampled micro-geometries (the third column) leads to significantly different appearances. Using the spatially varying multi-lobe BRDF f'_{low} without our scaling function $R = ST$ (the fourth column) performs better, but the results are still notably brighter than the references as changes of shadowing and masking are neglected. By applying a further correction using $R = ST$, our models well preserve the input appearances. Additionally, we compare our prefiltered models with those generated using LEADR [17], which are derived from single-lobe Beckmann NDFs and Smith's shadowing function. These models fail to reproduce the appearance of the input since the assumption of Gaussian surfaces does not hold (see the top insets in Figure 4.2(a, f)). Further, the single lobe representation used by LEADR also limits the result accuracy, especially for glossy base BRDFs.

Global illumination. The results in Figures 4.17 and 4.18 are rendered with full global illumination. In this case, directly using f_{orig} (the third column) still yields poor accuracy. Using multi-lobe BRDFs f'_{low} improves the glossy highlights quality but cannot fully capture the change of interreflections. Our full model with f'_{low} and the scaling functions $R_{\text{ir}} = S_{\text{ir}}T_{\text{ir}}$ resembles the input appearance accurately. To demonstrate the effectiveness and generality of our technique, we also show rendering results under different lighting and viewing configurations in Figure 4.17 and the accompanying video of the original publication.

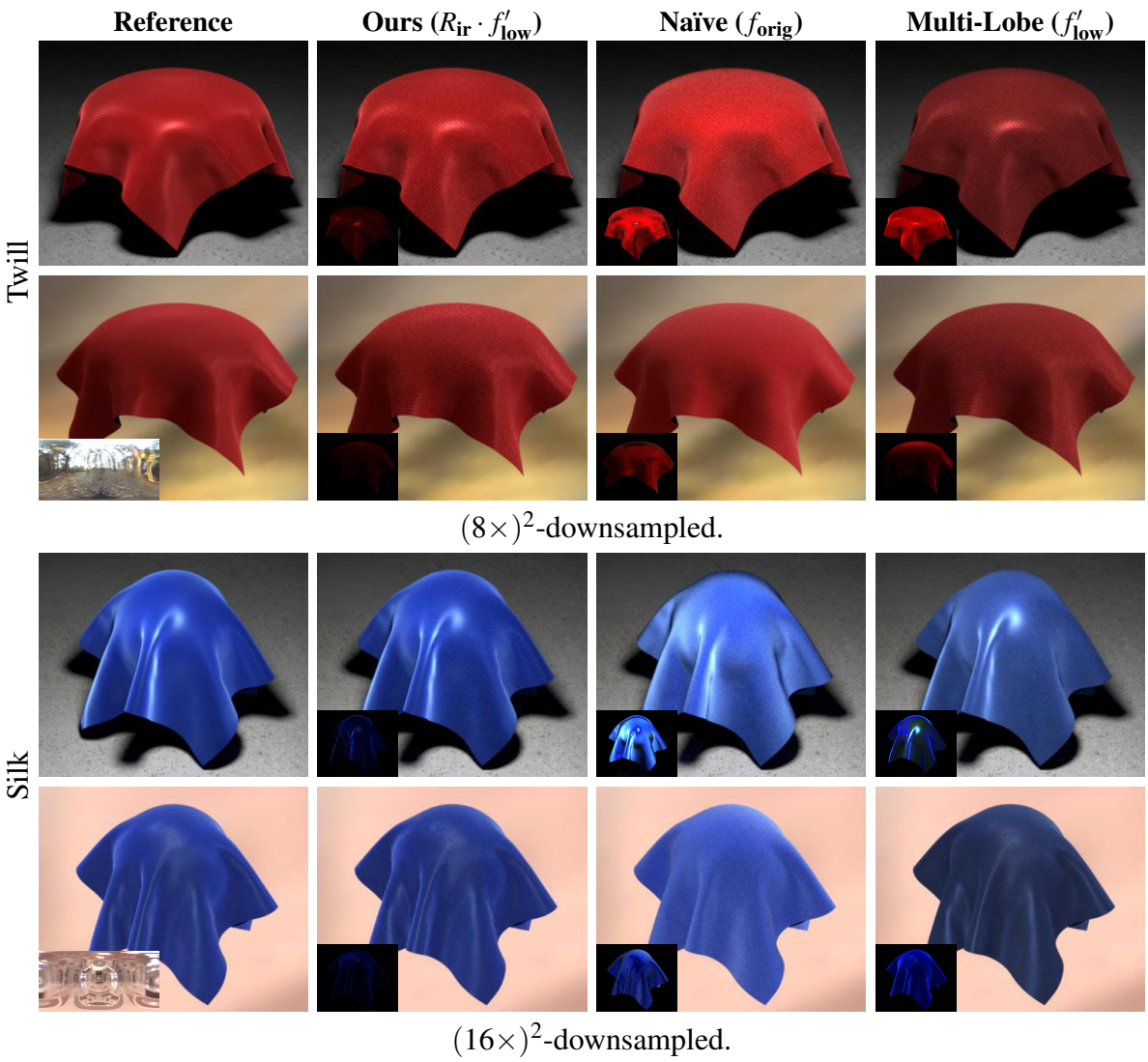


Figure 4.17. Global illumination results. The four columns on the left are rendered under local area lights, while the four columns on the right are rendered under environmental lightings (environment maps are visualized as insets of the reference images) from a different view. Our method is able to capture interreflections well, closely matching the reference. Naive downsampling and the multi-lobe BRDFs do not model interreflections and have significant errors, while LEADR does not perform well even for direct illumination and is not designed for interreflections (and is therefore not shown here).

Performance and storage. Tables 4.3 and 4.4 summarize the storage size and precomputation time of our prefiltered models, and Table 4.5 shows equal-time mean squared error (MSE) comparisons. Our prefiltered models can offer significant storage reduction at a single downsampling scale. Breaking down individual components of our model, the multi-lobe SVNDFs are

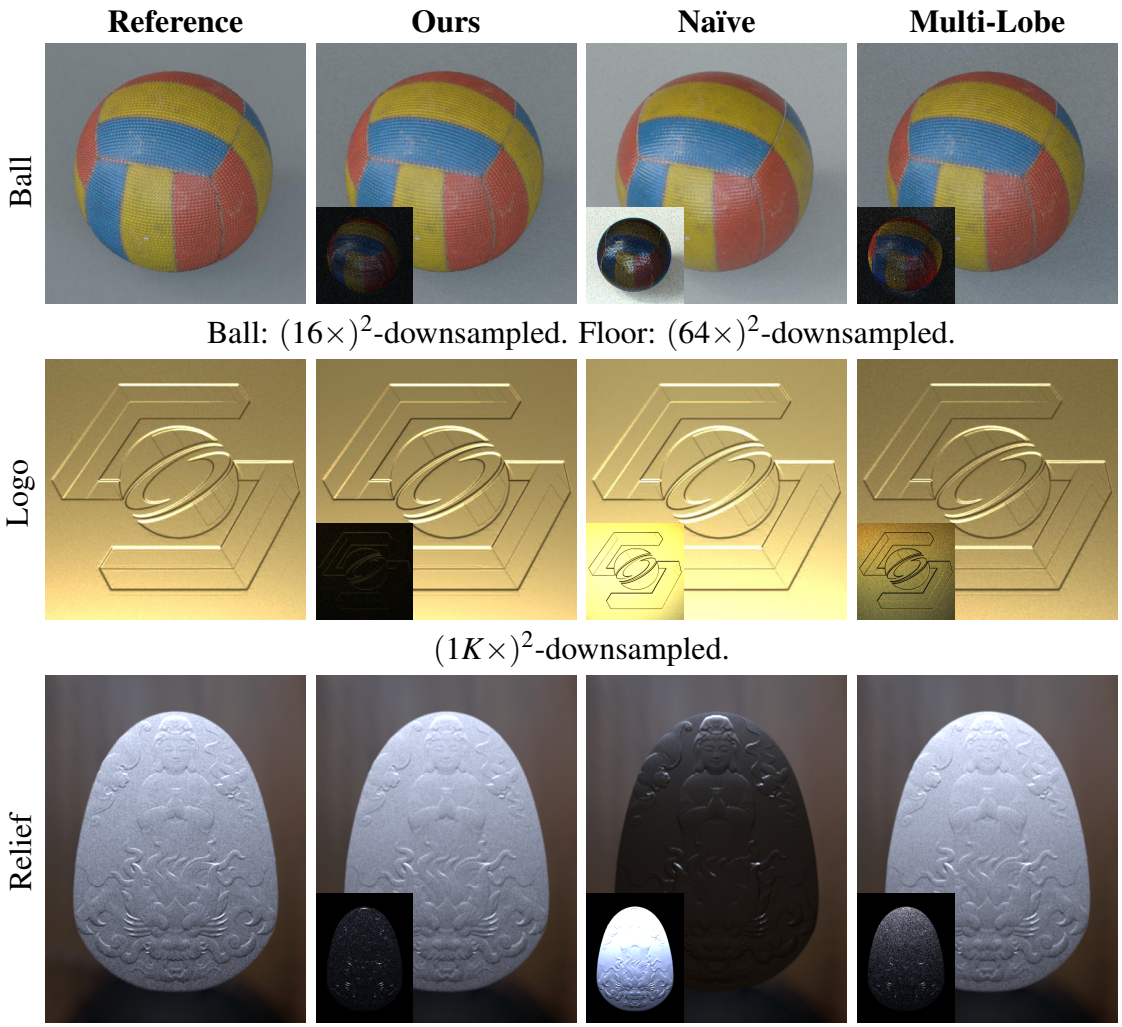


Figure 4.18. Global illumination results. We show results of more materials. Images rendered using our prefiltered models best match the references.

most storage-consuming (except "noise" and "logo" as they are greatly downsampled) while our scaling functions T and S only introduce minimal overhead.

Our model also benefits rendering performance by reducing ray tracing cost. We demonstrate this by rendering a few high-resolution models and their prefiltered counterparts (generated with our method) in equal-time and compare MSEs of the results (see Table 4.5). Using our prefiltered models leads to lower errors in all the scenes and anti-aliased renderings with better visual qualities (see Figure 4.2(c, e)). In the accompanying video of the original publication,

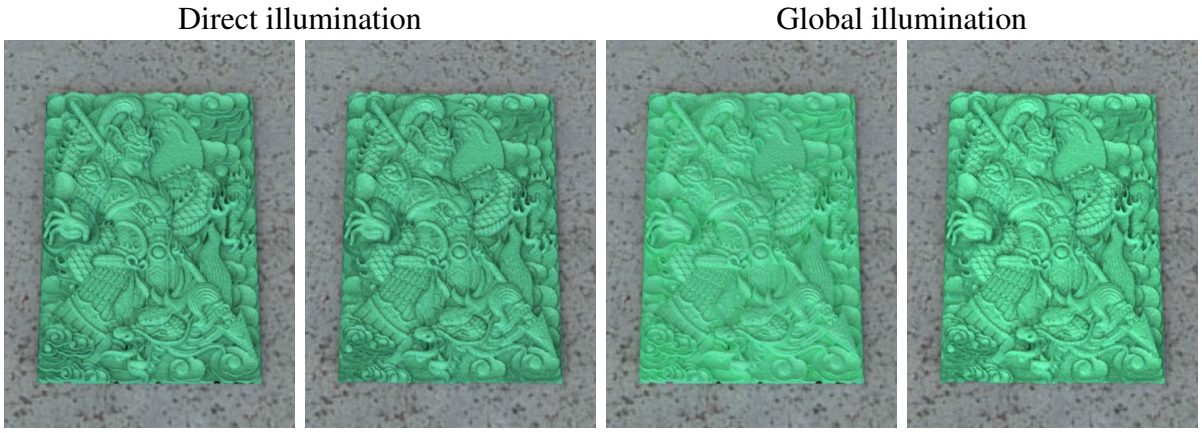


Figure 4.19. Failure case. Our method accurately handles direct illumination, but is unable to match the appearance with global illumination, since distant interreflections are not handled with sufficient accuracy.

we demonstrate that our prefiltered models preserve input appearance accurately and reduce flickering significantly.

4.8.3 Limitations and Future Work

Figure 4.19 shows a failure example where our method does not accurately reproduce the original appearance caused by interreflections. This "statue" scene contains a $(4K)^2$ displacement map and a moderately glossy base BRDF. We downsample it by $(16\times)^2$ and use a spatial scaling function of resolution $M = 1$. Compared to the "relief" scene, the vertical displacements in this scene are much greater. In this case, distant interreflections can only be partially handled by explicit path tracing. Since our method neglects these interreflections (for enabling efficient computation of S_{ir} and T_{ir}), the results have visible accuracy loss shown as darkening at the steep edges (see (b1) and (b2)). On the other hand, although our method produces inaccurate edge appearance, the overall surface reflectance (i.e. average image intensity) still matches the reference because the two effective BRDFs are matched by our scaling functions. For direct illumination results ((a1) and (a2)), the accuracy is not affected.

Since our method considers micro-geometries and base BRDFs jointly, which is neces-

sary for accurately handling complex light transport effects such as shadowing-masking and interreflections, changing the micro-geometry or base BRDF requires recomputing our prefiltered models. In the future, our technique may be combined with material editing techniques such as Hašan and Ramamoorthi [26] to allow changing the base BRDF with a single precomputation.

Interreflection is an effect that is well known to be challenging to analyze due to its high nonlinearity. A theoretical or data-driven analysis can be an interesting topic to explore. To balance accuracy and efficiency, we use relatively low resolutions for the spatial and angular scaling functions empirically. We leave how to determine the optimal resolutions as future work.

We assume the base BRDFs to be isotropic because the anisotropic micro-geometries are capable of reproducing anisotropic appearance. Using anisotropic base BRDFs would increase the model expressiveness but at the price of more expensive precomputation. We leave this extension for future exploration.

Additionally, as our technique only introduces minimal storage overhead, it may benefit real-time and interactive rendering applications.

4.9 Conclusion

We introduce a novel method to prefilter the surface reflectance generated by high-resolution displacement maps while accurately preserving the input appearance. Our prefiltered models leverage SVBRDFs expressed by a SVNDF coupled with a 6D scaling function that captures the change of shadowing-masking and interreflection effects caused by the downsampling of micro-geometries. We further introduce an efficient approach to factorize this 6D function into a 2D function of location and a 4D function of direction. Our models are capable of producing accurate appearance that matches the original one. In practice, our prefiltered models offer significant storage reduction and efficient anti-aliased rendering. Additionally, multiple models generated using our technique with varying downsampling scales can be combined to form a mipmap, allowing LoD rendering of detailed surfaces in an efficient and consistent manner.

4.10 Acknowledgements

This chapter is based on the material as it appears in ACM Transactions on Graphics, 2019 ("Accurate Appearance Preserving Prefiltering for Rendering Displacement-Mapped Surfaces", Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 5

Analytic Spherical Harmonic Gradients for Real-Time Rendering with Many Polygonal Area Lights

Next, we address the challenge of prefiltering complex lighting conditions, in particular, illumination from multiple polygonal area lights. The computational cost of accumulating contributions from many area lights is usually linear in the number of lights. To avoid the expensive computation at every shading point, we prefilter all the lights into a sparse 3D grid that stores spherical harmonic (SH) lighting coefficients and gradients. With the help of SH gradients, the SH lighting coefficients at any intermediate shading point can be accurately interpolated with a cost that is independent of the number of lights. Our prefiltering method builds on a novel analytic formula for SH gradients. This enables rendering scenes with hundreds of area lights using precomputed radiance transfer (PRT) in real-time.

5.1 Introduction

In this chapter, we present a novel technique to prefilter multiple polygonal area lights into a sparse 3D grid of spherical harmonic (SH) lighting coefficients and gradients, allowing scaling precomputed radiance transfer (PRT) [89] to hundreds of area lights. PRT and SH lighting enable dynamic low-frequency environments with realistic highlights and real-time shading, including soft shadows. Hence, they are widely used in real-time applications like

games and even in offline rendering [79]. However, the PRT method has often been limited to distant environment maps, since area light SH coefficients differ at each vertex on the object.

Recent work [102] has derived an analytic formula for SH coefficients for a uniform polygonal area light (building on earlier work on irradiance tensors [3, 91]), demonstrating area lights in PRT. A similar approach has also been applied to SH integrals for both offline and real-time rendering [5]. However, those methods require computing the SH coefficients at each integration point (each vertex in PRT), for each light, which precludes easily scaling to large numbers of lights.

An important observation is that the light field from area lights, and hence its SH coefficients, is smooth spatially. This suggests that the *spatial gradients of SH coefficients* are a critical quantity. Although gradients and differential methods have recently received significant attention [61, 117], there has so far been little previous work in analytically computing SH gradients (as opposed to SH coefficients). One reason may be the challenge of generalizing the SH coefficient derivation, which is already complex, with additional complexity from computing derivatives along each spatial direction.

In this work, we address this long-standing challenge, deriving an analytic formula for the spatial gradients of SH coefficients from a uniform polygonal area light. We further show how gradient-based interpolation can enable very sparse spatial sampling, and efficient prefiltering of multiple light sources with minimal overhead (see Figure 5.1). Our specific contributions include:

Derivation of analytic SH gradient formula. Our main contribution is the derivation of a novel analytic formula for SH gradients. While both area lights and spherical harmonics are widely used in rendering, to our knowledge, there has been no previous work on finding analytic SH gradients for them. This is a new result, which is a significant generalization of that for SH coefficients. We believe the result has implications for many problems in rendering and beyond. In particular, we show how to reduce the problem to a boundary integral (Section 5.4), where

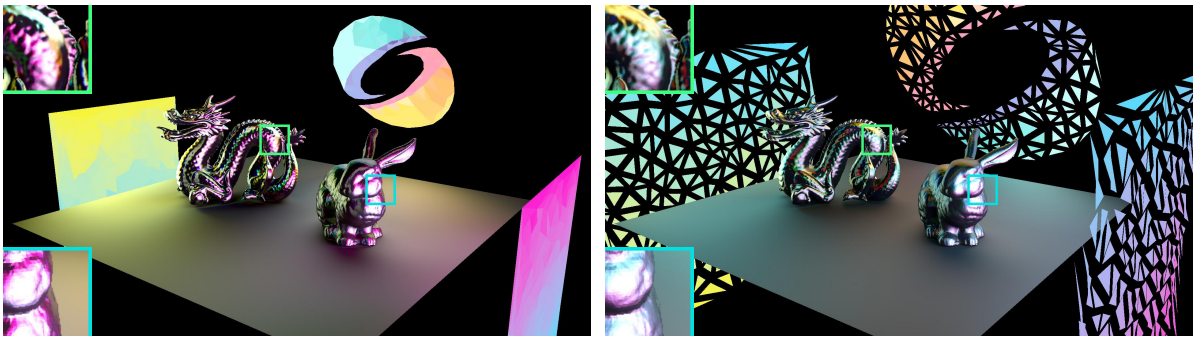


Figure 5.1. We present an analytic formula for spherical harmonic (SH) gradients from uniform polygonal area lights, and show how this new theoretical result enables scaling Precomputed Radiance Transfer (PRT) to hundreds of area lights. We first compute the lighting SH coefficients and gradients on a sparse 3D grid. To evaluate SH coefficients for any intermediate point (vertex), we exploit SH gradients and use accurate Hermite interpolation. Here we render a glossy scene with 713 polygonal (triangular) lights and 1.3M polygons at 36fps. Each light transforms independently (in terms of color, location, orientation etc.), enabling the appearance of textured lights or more complex patterns, and causing significant changes in glossy highlights (compare left and right images).

a key term can be reduced to an earlier SH coefficient recurrence (Section 5.5). Our practical Algorithm 2 in Section 5.6.1 is simple, requiring only a few simple modifications to existing code for computing analytic SH coefficients.

Gradient-based interpolation. We demonstrate gradient-based interpolation from a sparse set of samples. An overview of our method is in Algorithm 1 in Section 5.6. We develop a Hermite cubic interpolant that is consistent with the analytic gradients (Section 5.6.2 and Algorithm 4), and more accurate than previous Taylor-series based numerical approaches [1]. Even for rendering with one area light source, we demonstrate a $2\times$ speedup over explicit analytic computation of SH coefficients at each vertex. Our major benefit is for handling multiple area lights, where we can linearly accumulate SH coefficients and gradients for all lights on a sparse grid with minimal overhead, followed by gradient-based interpolation.

Efficient real-time prefiltering of multiple area lights. We implement our approach within a real-time PRT system. We can prefilter *hundreds of uniform polygonal area lights* into SH coefficients and gradients in *real-time*, which was not previously possible (Section 5.7,

Figures 5.1, 5.10, and 5.11). This approach also enables easily breaking a high-resolution textured light source into uniform polygonal luminaires, which can be handled with our method.

5.2 Related Work

PRT and spherical harmonics. SH have been widely used in both real-time and offline rendering, going back to the work by Cabral et al. [8]. In particular, they have widely been used in practice for PRT [89], enabling soft shadows and other light transport effects. Approaches based on all-frequency relighting [71] can be more accurate but have not gained widespread practical adoption because of the high precomputation and storage costs. Other analytic methods for area lights, such as the work by Heitz et al. [30], do not consider shadows. There have been many subsequent developments in PRT; we describe only the closest previous work and refer readers to a survey [60, 82] for a more thorough introduction.

Annen et al. [1] proposed spherical harmonic gradients for mid-range illumination, improving on simply interpolating a small number of locations for incident illumination [89]. However, they did not derive an analytic gradient formula, and required 2D numerical integration over the area light source, with complexity still scaling linearly with the number of lights. In Section 5.4 and Appendix C.2, we show that their result is essentially a different parameterization; our approach enables reduction to a boundary integral with an analytic form.

Zhou et al. [122] introduced dynamic scenes and near-field lights for all-frequency relighting. Using gradients enables sparser representations, which in turn enable scaling with only a small overhead to hundreds of lights, which was not previously possible. Since our algorithm only pertains to the lighting, other benefits such as dynamic scenes, or any other PRT transport algorithm, can easily be included. Other all-frequency area light methods include the wavelet propagation approach [93] which also handles texture, but the results require approximations and some operations like out-of-plane light rotation are not permitted. Their method is again limited to a single or small number of lights.

Ren et al. [85] break lights and geometry into spheres and use spherical harmonic exponentiation to enable real-time soft shadows in more complex dynamic scenes. However, the spherical approximation is not generally suitable for planar area lights (even using multiple spheres is a poor approximation of a planar surface from all directions). Moreover, analytic formulae are provided only for sphere lights. Note that none of the above-mentioned papers compute analytic spherical harmonic gradients, and this computation may also benefit these approaches in the future.

Illumination gradients. Irradiance and radiance caching [54, 55, 104] are important techniques for global illumination. Greger et al. [23] introduced the irradiance volume to precompute and store the irradiance distribution function in a volumetric grid. Shading at arbitrary points is computed by trilinearly interpolating from the irradiance volume. In our work, we precompute and store SH coefficients and gradients in the grid, and perform more accurate Hermite interpolation using the gradients.

A series of works [45, 47, 52, 53, 103] exploit irradiance and radiance gradients to enable sparse sampling of (ir)radiance caching points and accurate value interpolation. They use Monte Carlo integration to evaluate those gradients numerically. In contrast, we derive analytic formulae for SH gradients. More recently, second-order derivatives (Hessians) have been used for error control of the first-order approximation using gradients [46, 69, 86]. It is worth exploring analytic forms of higher order derivatives in the future.

Holzschuch and Sillion [33] developed analytic radiosity gradients with constant and linear emitters. They used Stokes' theorem to separate the radiosity gradient into two parts: a contour integral and a surface integral. We achieve similar results using the Reynolds transport theorem [57]. Their follow-up work [34] presented analytic second-order derivatives of the radiosity function, enabling error bounding for hierarchical radiosity algorithms. However, their analysis of the radiosity method only handles diffuse surface patches, while our SH gradients

apply to incident lighting and can also be used for glossy surfaces.

Differentiable rendering. Building on early approaches to gradient light transport [2, 84], recent efforts enable differentiating paths including shadows [61] and radiative transport [117], with different parameterizations [65]. Our method can also be viewed as a differentiable rendering technique, and indeed makes use of the Reynolds transport theorem [57], also used in the work by Zhang et al. [117]. However our goals are very different: we compute SH gradients for PRT, deriving a novel analytic formula. Insights from our work can be relevant in the future for differentiable rendering and machine learning applications [63].

Numerical/Automatic differentiation. For gradient computations, it is possible to use numerical finite differencing. However, this requires finding the appropriate step size, and can be noisy and inefficient, as noted in the papers above. It is also possible to use automatic differentiation (for example, as used in the work by Li et al. [62]). However, the results are not optimized, and are less efficient than our analytic gradients. Moreover, we present an explicit analytical derivation, with novel insights, which cannot be achieved with automatic differentiation.

5.3 Preliminaries

We first introduce basic background on PRT, zonal harmonic factorization, area lights and differentiating integrals.

5.3.1 Reflection Equation and PRT

The simplest version of precomputed radiance transfer (PRT) tries to solve the reflection equation at spatial position $\boldsymbol{x}$,

$$B(\boldsymbol{x}) = \int_{\mathcal{S}^2} L_i(\boldsymbol{x}, \boldsymbol{\omega}_i) T(\boldsymbol{x}, \boldsymbol{\omega}_i) \, \mathrm{d}\boldsymbol{\omega}_i, \tag{5.1}$$

where $B(\boldsymbol{x})$ is the reflected radiance or image intensity,[1] $L_i(\boldsymbol{x}, \boldsymbol{\omega}_i)$ is the incoming lighting at $\boldsymbol{x}$ from direction $\boldsymbol{\omega}_i$, and we integrate over the sphere of incoming directions. $T(\boldsymbol{x}, \boldsymbol{\omega}_i)$ is the transport function which is precomputed at each vertex in PRT, and encapsulates the BRDF, cosine term and visibility.

In this chapter, our focus is on lighting, i.e., efficient SH projections of L_i, rather than the transport T. Any general PRT method can be used to handle the transport function without modifications to the relevant code. In general, L_i and T are expanded in (real) spherical harmonics $Y_{lm}(\boldsymbol{x})$, which are orthonormal basis functions on the unit sphere. Therefore, the integral reduces to a simple summation,

$$B(\boldsymbol{x}) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} L_{lm}(\boldsymbol{x}) T_{lm}(\boldsymbol{x}), \tag{5.2}$$

where L_{lm} and T_{lm} are spherical harmonic coefficients for the lighting and transport respectively (with $L = \sum_{l,m} L_{lm} Y_{lm}$ and $T = \sum_{l,m} T_{lm} Y_{lm}$). We typically consider $l_{\max} = 8$ in this chapter, involving 81 SH terms; the summation can be computed at each vertex $\boldsymbol{x}$ as a dot-product of lighting and transport coefficient vectors in graphics hardware for each color channel.

In the original PRT formulation for distant environment maps, lighting is independent of spatial position $\boldsymbol{x}$ and the lighting coefficients L_{lm} can be computed once for each frame, while transport coefficients $T_{lm}(\boldsymbol{x})$ are precomputed and stored. In our case, for near-field area lighting, $L_{lm}(\boldsymbol{x})$ changes at each spatial location.

5.3.2 Zonal Harmonic Factorization

Zonal harmonics (ZH) are a subset of SH basis functions for $m = 0$ (see definition in Section 2.2.1). As pointed out by Nowrouzezahrai et al. [75], any SH basis function Y_{lm} can be

[1] For notational simplicity, we assume diffuse reflections and drop dependence of reflected direction $\boldsymbol{\omega}_o$ in B and T. In general, we can also handle non-diffuse reflections $B(\boldsymbol{x}, \boldsymbol{\omega}_o)$ using a glossy PRT extension (see Figure 5.1). Since we focus only on lighting L_i, any PRT framework can be supported, including interreflections and dynamic scenes.

sparsely factorized into a weighted sum of rotated ZH basis functions,

$$Y_{lm}(\boldsymbol{\omega}) = \sum_j \alpha_{l,j}^m Y_{l0}(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_{l,j}), \tag{5.3}$$

where $\boldsymbol{\omega}_{l,j}$ represents the central direction of each rotated ZH lobe and $\alpha_{l,j}^m$ is its corresponding weight. Theoretically, to represent each band-l SH basis function, $2l+1$ rotated ZH lobes are required. However in practice, the ZH lobes can be shared across all bands (Appendix C.1), resulting in a sparse weight matrix $\alpha_{l,j}^m$ and faster SH rotation.

5.3.3 Analytic Spherical Harmonic Coefficients

Given a polygonal light source A with unit intensity, we denote its projection onto a unit sphere centered at the shading point $\boldsymbol{x}$ as $Q(\boldsymbol{x})$. The SH coefficients of this area light source with respect to $\boldsymbol{x}$ are given by integrating the SH basis functions over Q,

$$L_{lm}(\boldsymbol{x}) = \int_{Q(\boldsymbol{x})} Y_{lm}(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega}. \tag{5.4}$$

By applying the zonal harmonic factorization (Equation (5.3)) to Y_{lm}, the SH coefficients can be rewritten as

$$\begin{aligned} L_{lm}(\boldsymbol{x}) &= \int_{Q(\boldsymbol{x})} \left(\sum_j \alpha_{l,j}^m Y_{l0}(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_{l,j}) \right) \mathrm{d}\boldsymbol{\omega} \\ &= \sum_j \alpha_{l,j}^m \underbrace{\int_{Q(\boldsymbol{x})} Y_{l0}(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_{l,j})\, \mathrm{d}\boldsymbol{\omega}}_{=: L_{l,j}(\boldsymbol{x})}. \end{aligned} \tag{5.5}$$

Therefore, computing $L_{lm}(\boldsymbol{x})$ boils down to computing the ZH coefficients $L_{l,j}(\boldsymbol{x})$. Belcour et al. [5] further represented the ZH coefficient as the sum of cosine-power integrals and found analytic solutions for these integrals over spherical polygons. Wang and Ramamoorthi [102] derived analytic ZH coefficients by applying Stokes' Theorem to convert surface integrals to boundary integrals. The boundary integrals are further solved using the recurrence relations

of Legendre polynomials, which are also used in our derivation. However, our approach is different, in terms of finding the SH gradients, which we reduce to a novel boundary integral by differentiating the surface integral for ZH coefficients. A crucial insight helping us to find the analytic formula is in reducing a key term to the earlier ZH coefficient recurrence, which involves minimal overhead to evaluate coefficients and gradients jointly.

5.3.4 Differentiating Integrals

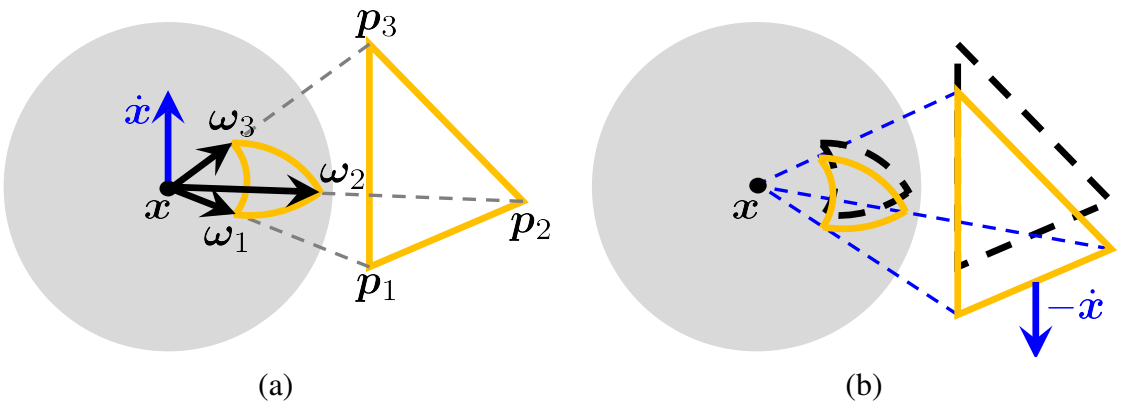
In this work, our goal is to find the SH spatial gradients $\nabla_{\boldsymbol{x}} L_{lm}(\boldsymbol{x})$, which we will often denote simply as $\nabla L_{lm}(\boldsymbol{x})$. This reduces to differentiating the integral on the right-hand side (RHS) of Equation (5.4). To this end, we leverage the Reynolds transport theorem which originates in fluid mechanics [57] and generalizes the Leibniz integral rule for differentiation under the integral operator.

Let $\Omega(\pi)$ be an n-dimensional manifold parameterized by π. We are interested in differentiating the integration of a function f over the region $\Omega(\pi)$. The partial derivative with respect to π can be expressed as

$$\partial_\pi \left(\int_{\Omega(\pi)} f \, \mathrm{d}\Omega(\pi) \right) = \int_{\Omega(\pi)} \dot{f} \, \mathrm{d}\Omega(\pi) + \int_{\partial\Omega(\pi)} \langle \boldsymbol{n}, \dot{\boldsymbol{x}} \rangle f \, \mathrm{d}\partial\Omega(\pi), \tag{5.6}$$

where $\partial_\pi := \frac{\partial}{\partial_\pi}$ and $\dot{f} := \partial_\pi f$. The differentiation result has two parts. The first one is an integral on the original domain $\Omega(\pi)$. The other one is a boundary integral on the $(n-1)$-dimensional region $\partial\Omega(\pi)$. For its integrand, we define $\dot{\boldsymbol{x}} := \partial_\pi \boldsymbol{x}$, $\boldsymbol{n}$ is the normal direction at each $\boldsymbol{x} \in \partial\Omega(\pi)$ and points towards the exterior by convention, and $\langle \cdot, \cdot \rangle$ is the dot product of two vectors.

Recent work in differentiable rendering [61, 117] have already shown the significance of the boundary integral for correctly evaluating geometric derivatives. Later in Sections 5.4 and 5.5, we will see the boundary integral also leads to significant formula simplification and enables the analytic derivation.



(a) (b)

Figure 5.2. (a) Illustration of the spherical projection for a triangle light. (b) When the shading point $\boldsymbol{x}$ moves by $\dot{\boldsymbol{x}}$ (equivalent to the area light moves by $-\dot{\boldsymbol{x}}$), the projected spherical polygon also changes accordingly.

5.4 Differentiating Spherical Harmonic Coefficients

In this section, we will use the Reynolds transport theorem to differentiate the SH coefficients for area lights, showing that SH gradients can be computed from boundary integrals. In Section 5.5 we will further derive our key result, an analytic formula. In Section 5.6 we develop our SH gradient (jointly with SH coefficients) evaluation algorithm and the gradient-based interpolation method, showing how to handle multiple area light sources. While the derivation is somewhat involved, the actual algorithm involves only a few simple modifications to previous code for SH coefficients. Readers interested primarily in implementation may wish to first browse Section 5.6 and Algorithms 1 and 2.

Let $A = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N)$ be a uniform polygonal light source with N points in $\mathbb{R}^3$ and $\boldsymbol{x}$ be the shading point where we want to evaluate the incident radiance (see Figure 5.2(a)). Note that $Q(\boldsymbol{x}) = (\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_N)$ is a spherical polygon obtained by projecting A onto the unit sphere $\mathcal{S}^2$ centered at $\boldsymbol{x}$ (see Figure 5.2(a)), where $\boldsymbol{\omega}_i = \frac{\boldsymbol{p}_i - \boldsymbol{x}}{\|\boldsymbol{p}_i - \boldsymbol{x}\|}$. Both $Q(\boldsymbol{x})$ and its boundary (which consists of arcs on $\mathcal{S}^2$) $\partial Q(\boldsymbol{x}) = \{\widehat{\boldsymbol{\omega}_i \boldsymbol{\omega}_{i+1}} \,|\, i = 1, \ldots, N\}$ may vary with $\boldsymbol{x}$.

5.4.1 Spherical Harmonic Gradient

Given a shading point $\boldsymbol{x} = (x, y, z)$, we define the SH gradient as the spatial gradients with respect to $\boldsymbol{x}$,

$$\nabla L_{lm}(\boldsymbol{x}) = (\partial_x L_{lm}(\boldsymbol{x}), \partial_y L_{lm}(\boldsymbol{x}), \partial_z L_{lm}(\boldsymbol{x})). \tag{5.7}$$

Without loss of generality, we will focus on one of the spatial partial derivatives $\partial_z L_{lm}(\boldsymbol{x})$ in the following derivations. The other gradient components can be evaluated in the same way.

To evaluate the SH coefficient in Equation (5.4), the SH basis function is integrated over a varying domain $Q(\boldsymbol{x})$ as the shading point $\boldsymbol{x}$ changes. By applying the Reynolds transport theorem to the right-hand side (RHS) of Equation (5.4), we can write the partial derivative as

$$\partial_z \int_{Q(\boldsymbol{x})} Y_{lm}(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega} = \int_{Q(\boldsymbol{x})} \partial_z [Y_{lm}(\boldsymbol{\omega})]\,\mathrm{d}\boldsymbol{\omega} + \int_{\partial Q(\boldsymbol{x})} \langle \boldsymbol{n}_\perp, \dot{\boldsymbol{\omega}} \rangle Y_{lm}(\boldsymbol{\omega})\,\mathrm{d}\ell(\boldsymbol{\omega}). \tag{5.8}$$

The first integral on the RHS vanishes, because the SH basis function is independent of the shading point position so $\partial_z[Y_{lm}(\boldsymbol{\omega})] = 0$.

The second integral is due to the moving boundary $\partial Q(\boldsymbol{x})$ as $\boldsymbol{x}$ varies (see Figure 5.2(b)). For every $\boldsymbol{\omega} \in \partial Q(\boldsymbol{x})$, $\mathrm{d}\ell(\boldsymbol{\omega})$ represents the arc length measure. The normal vector $\boldsymbol{n}_\perp$ is in the tangent space of $\boldsymbol{\omega} \in \mathcal{S}^2$ and perpendicular to the boundary curve. We denote $\dot{\boldsymbol{\omega}}$ as the change rate of the boundary location $\boldsymbol{\omega}$ with respect to z, i.e. $\dot{\boldsymbol{\omega}} = \partial_z \boldsymbol{\omega}$. Let $\boldsymbol{y} \in \partial A$ be a point on the polygonal light boundary. Further, the shading point $\boldsymbol{x}$ has a change rate $\dot{\boldsymbol{x}} = (0, 0, 1)$. We can establish the following relation between the change rates (see Figure 5.3),

$$\dot{\boldsymbol{\omega}} = \partial_z \left(\frac{\boldsymbol{y} - \boldsymbol{x}}{\|\boldsymbol{y} - \boldsymbol{x}\|} \right) = \frac{-\dot{\boldsymbol{x}}}{\|\boldsymbol{y} - \boldsymbol{x}\|} - \boldsymbol{\omega} \left\langle \boldsymbol{\omega}, \frac{-\dot{\boldsymbol{x}}}{\|\boldsymbol{y} - \boldsymbol{x}\|} \right\rangle. \tag{5.9}$$

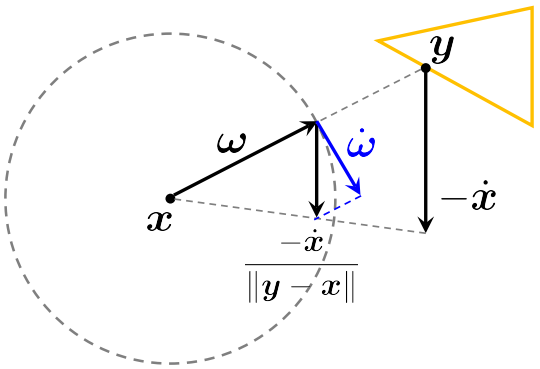


Figure 5.3. Evaluating the change rate of $\boldsymbol{\omega}$. We first project $-\dot{\boldsymbol{x}}$ to the unit sphere, then obtain $\dot{\boldsymbol{\omega}}$ by extracting the component of $\frac{-\dot{\boldsymbol{x}}}{\|\boldsymbol{y}-\boldsymbol{x}\|}$ that is perpendicular to $\boldsymbol{\omega}$.

5.4.2 Reduction to Edge/Arc Integrals

The previous subsection shows that the spatial partial derivative can be simplified as

$$\partial_z L_{lm}(\boldsymbol{x}) = \int_{\partial Q(\boldsymbol{x})} \langle \boldsymbol{n}_\perp, \dot{\boldsymbol{\omega}} \rangle Y_{lm}(\boldsymbol{\omega}) \, \mathrm{d}\ell(\boldsymbol{\omega}), \tag{5.10}$$

which is a 1D integral on the spherical polygon edges.

We now decompose the boundary integral into a sum of arc integrals for each arc $\widehat{\boldsymbol{\omega}_i \boldsymbol{\omega}_{i+1}} \in \partial Q(\boldsymbol{x})$,

$$\partial_z L_{lm}(\boldsymbol{x}) = \sum_{i=1}^{N} \underbrace{\int_{\widehat{\boldsymbol{\omega}_i \boldsymbol{\omega}_{i+1}}} \langle \boldsymbol{n}_i, \dot{\boldsymbol{\omega}} \rangle Y_{lm}(\boldsymbol{\omega}) \, \mathrm{d}\ell(\boldsymbol{\omega})}_{=: G_{lm}^{(i)}}. \tag{5.11}$$

For every $\boldsymbol{\omega} \in \widehat{\boldsymbol{\omega}_i \boldsymbol{\omega}_{i+1}}$, it has the same normal direction

$$\boldsymbol{n}_i = \frac{\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i+1}}{\|\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i+1}\|}. \tag{5.12}$$

Further, the edge integral can be parameterized with the radian angle $t \in [0, T_i]$ as

$$\boldsymbol{\omega}(t) = \boldsymbol{\omega}_i \cos t + \boldsymbol{\lambda} \sin t, \tag{5.13}$$

where $T_i = \arccos(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_{i+1})$ and $\boldsymbol{\lambda} = \boldsymbol{n}_i \times \boldsymbol{\omega}_i$. The direction change rate $\dot{\boldsymbol{\omega}}(t)$ can be evaluated using Equation (5.9). The arc length measure $\mathrm{d}\ell(\boldsymbol{\omega})$ is equal to $\mathrm{d}t$ since the sphere radius is one. In summary, we can rewrite the arc integrals in Equation (5.11) as simple 1D integrals:

$$\boxed{G_{lm}^{(i)} = \int_0^{T_i} \langle \boldsymbol{n}_i, \dot{\boldsymbol{\omega}}(t) \rangle Y_{lm}(\boldsymbol{\omega}(t)) \, \mathrm{d}t.} \tag{5.14}$$

At this point, it would be possibly to simply evaluate $G_{lm}^{(i)}$ efficiently using 1D numerical quadrature rules.[2] However, we will go even further, deriving a fully analytic recurrence formula in Section 5.5.

Discussion. The reduction of area light computations to edge integrals over the bounding arcs is common, but previous work has used Stokes' theorem for polynomial or spherical harmonic coefficients [91, 102]. Our use of the Reynolds transport theorem to reduce the *gradients* to boundary integrals is a novel approach, to the best of our knowledge.

Annen et al. [1] developed a semi-analytic solution to SH gradients. We show in Appendix C.2 that their results can also be derived from the Reynolds transport theorem, but using a different parameterization. In their case, the integration domain is independent of the varying parameters. As a result, the boundary integral (second integral on the RHS of Equation (5.8)) becomes zero after differentiating with the Reynolds transport theorem. On the other hand, we use a parameterization such that the integration domain varies while the integrand is independent. Therefore, the integral of the differentiated quantity (first integral on the RHS of Equation (5.8)) is zero.

Although both methods result in equivalent solutions, they have different algorithmic implications. In the work by Annen et al. [1], they need to numerically integrate functions in 2D, even though the integrand can be evaluated analytically or by automatic differentiation. On the other hand, our parameterization results in dimension reduction. The 1D integrals can

[2] Note that the integrand is smooth and low-dimensional (1D), so Monte Carlo sampling is not required; a standard integration rule like Simpson's or a higher-order quadrature scheme could be employed. This may be desirable in some applications.

not only be evaluated numerically in a more efficient way, but also lead to analytic solutions. Different applications may prefer specific parameterizations. For example, in differentiable rendering [61, 65, 117], one wants to avoid the boundary integration as much as possible, because the gradient estimation requires additional effort for edge sampling. Further, the edge integrals they evaluated are too complex to have analytic solutions. In contrast, we prefer to reduce SH gradients to edge integrals. The dimension reduction on the integral domain makes the analytic derivation much easier.

5.5 Analytic Formula

In this section, we will show how to solve SH gradients analytically. First, we use ZH factorization [75], see Equations (5.3) and (5.5), to rewrite Equation (5.14) in terms of ZH integrals (we focus on one edge in the following derivations),

$$G_{lm}^{(i)} = \sum_j \alpha_{l,j}^m \underbrace{\int_0^{T_i} \langle \boldsymbol{n}_i, \dot{\boldsymbol{\omega}}(t) \rangle Y_{l0}(\boldsymbol{\omega}(t) \cdot \boldsymbol{\omega}_{l,j}) \, \mathrm{d}t}_{=: G_{l,j}^{(i)}} . \tag{5.15}$$

The weights $\alpha_{l,j}^m$ and central directions $\boldsymbol{\omega}_{l,j}$ of the ZH lobes can be precomputed. Now, the problem reduces to how to evaluate the integral $G_{l,j}^{(i)}$ analytically.

The input of our method includes the edge endpoints $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$, the shading point $\boldsymbol{x}$, its change rate $\dot{\boldsymbol{x}}$ that equals $(0,0,1)$ when differentiating with respect to z, and the central direction $\boldsymbol{\omega}_{l,j}$ of the j-th ZH lobe. Briefly, the derivation involves simplifying the integrand, rearranging terms and reducing to the known recurrence relations of Legendre polynomials.

5.5.1 Solving for $G_{l,j}^{(i)}$

Transforming to local frame. We seek to represent the integrand of $G_{l,j}^{(i)}$ by a function of t, i.e., $g(t) = \langle \boldsymbol{n}_i, \dot{\boldsymbol{\omega}}(t) \rangle Y_{l0}(\boldsymbol{\omega}(t) \cdot \boldsymbol{\omega}_{l,j})$, and simplify it as much as possible. First, we translate the edge $\overline{\boldsymbol{p}_i \boldsymbol{p}_{i+1}}$ by $-\boldsymbol{x}$ so that the shading point is at the origin. We denote the distances from the

shading point to the edge endpoints as $\ell_i = \|\boldsymbol{p}_i - \boldsymbol{x}\|$ and $\ell_{i+1} = \|\boldsymbol{p}_{i+1} - \boldsymbol{x}\|$. The arc $\widehat{\boldsymbol{\omega}_i \boldsymbol{\omega}_{i+1}}$ is represented by two unit vectors $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_{i+1}$. Then, we build a local frame $(\boldsymbol{\omega}_i, \boldsymbol{\lambda}_i, \boldsymbol{n}_i)$, where $\boldsymbol{n}_i$ and $\boldsymbol{\lambda}_i$ are defined in Equations (5.12) and (5.13). We transform all the related vectors $\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i+1}$, and $\boldsymbol{\omega}_{l,j}$ into this local frame by a rotation operator $\mathcal{R}(\boldsymbol{u}) = (\boldsymbol{u} \cdot \boldsymbol{\omega}_i, \boldsymbol{u} \cdot \boldsymbol{\lambda}_i, \boldsymbol{u} \cdot \boldsymbol{n}_i)$. One benefit is that expressions of $\boldsymbol{\omega}(t)$ are simpler after rotation: $\mathcal{R}(\boldsymbol{\omega}(t)) = (\cos t, \sin t, 0)$, because the edge is completely in the xy-plane and $\boldsymbol{\omega}_i$ aligns with the x-axis. Moreover, the function value $g(t)$ is unchanged since the dot product is invariant under rotation.

Simplifying $\langle \boldsymbol{n}_i, \dot{\boldsymbol{\omega}}(t) \rangle$. To evaluate this term, we can expand and simplify it using Equation (5.9):

$$\langle \boldsymbol{n}_i, \dot{\boldsymbol{\omega}}(t) \rangle = \left\langle \boldsymbol{n}_i, \frac{-\dot{\boldsymbol{x}}}{\|\boldsymbol{y}(t) - \boldsymbol{x}\|} \right\rangle - \langle \boldsymbol{n}_i, \boldsymbol{\omega}(t) \rangle \left\langle \boldsymbol{\omega}(t), \frac{-\dot{\boldsymbol{x}}}{\|\boldsymbol{y}(t) - \boldsymbol{x}\|} \right\rangle = -\frac{1}{\|\boldsymbol{y}(t) - \boldsymbol{x}\|} \langle \boldsymbol{n}_i, \dot{\boldsymbol{x}} \rangle, \quad (5.16)$$

because $\boldsymbol{n}_i$ is perpendicular to $\boldsymbol{\omega}(t)$. Assuming $\boldsymbol{n}_i = (n_x, n_y, n_z)$, we know that $\langle \boldsymbol{n}_i, \dot{\boldsymbol{x}} \rangle = n_z$. Then, it remains to find $\ell(t) = \|\boldsymbol{y}(t) - \boldsymbol{x}\|$. We denote $\boldsymbol{y}(t) = \boldsymbol{x} + \ell(t)\boldsymbol{\omega}(t)$ as the intersection point of $\overline{\boldsymbol{p}_i \boldsymbol{p}_{i+1}}$ and the ray with direction $\boldsymbol{\omega}(t)$ starting at $\boldsymbol{x}$. The solution can be found by solving a linear system,

$$\ell(t) = \left(\frac{\sin t}{\ell_{i+1}} - \frac{\sin(t - T_i)}{\ell_i} \right)^{-1} \sin T_i. \quad (5.17)$$

Recall that ℓ_i and ℓ_{i+1} are the distances to area light vertices $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ respectively, from the shading point $\boldsymbol{x}$ (corresponding to $\ell(0)$ and $\ell(T_i)$). We provide the detailed derivation in Appendix C.3.1.

To sum up, Equation (5.16) can be written as

$$\langle \boldsymbol{n}_i, \dot{\boldsymbol{\omega}}(t) \rangle = -\frac{n_z}{\sin T_i} \left(\frac{\sin t}{\ell_{i+1}} - \frac{\sin(t - T_i)}{\ell_i} \right). \quad (5.18)$$

Simplifying $Y_{l0}(\boldsymbol{\omega}(t) \cdot \boldsymbol{\omega}_{l,j})$. Given a precomputed central direction $\boldsymbol{\omega}_{l,j}$, we denote the

vector after rotation as $\mathcal{R}(\boldsymbol{\omega}_{l,j}) = (c_x, c_y, c_z)$. The ZH basis function can be written as

$$Y_{l0}(\mathcal{R}(\boldsymbol{\omega}(t)) \cdot \mathcal{R}(\boldsymbol{\omega}_{l,j})) = K_l P_l(c_x \cos t + c_y \sin t). \tag{5.19}$$

Finally, plugging Equations (5.18) and (5.19) back in Equation (5.15), we have

$$\begin{aligned} G_{l,j}^{(i)} = & \frac{n_z K_l}{\ell_i \sin T_i} \int_0^{T_i} \sin(t - T_i) P_l(c_x \cos t + c_y \sin t)\, \mathrm{d}t - \\ & \frac{n_z K_l}{\ell_{i+1} \sin T_i} \int_0^{T_i} \sin(t) P_l(c_x \cos t + c_y \sin t)\, \mathrm{d}t. \end{aligned} \tag{5.20}$$

Rearranging terms. Despite our simplifications to the integrand, it remains nontrivial to evaluate the integrals in Equation (5.20) analytically. Fortunately, for $h(t) = c_x \cos t + c_y \sin t$, the integral $\int h P_l(h)\, \mathrm{d}t$ does have a closed-form solution, which can be derived using integration by parts [102]. Therefore, our goal is to rearrange the integrand so that it reduces to this integral.

We first combine the linear combination of sine and cosine waves to a single sine wave with a scaled amplitude A and a phase shift T',

$$h(t) = c_x \cos t + c_y \sin t = A \sin(t + T'), \tag{5.21}$$

where $A = \sqrt{c_x^2 + c_y^2}$ and $T' = \arctan(c_x / c_y)$. Using trigonometric identities, the first integral on the RHS of Equation (5.20) can be reformulated as

$$\frac{\cos(T_i + T')}{A} C_l - \frac{\sin(T_i + T')}{A} E_l, \tag{5.22}$$

where $C_l = \int_0^{T_i} h P_l(h)\, \mathrm{d}t$ and $E_l = \int_0^{T_i} (\frac{\mathrm{d}}{\mathrm{d}t} h) P_l(h)\, \mathrm{d}t$. We provide the derivation details in Appendix C.3.2. The second integral can be solved in the same way. Finally, Equation (5.20) can

be simplified as

$$\boxed{\begin{aligned} G_{l,j}^{(i)} =& \frac{n_z K_l}{A\ell_i \sin T_i}(C_l \cos(T_i + T') - E_l \sin(T_i + T')) - \\ & \frac{n_z K_l}{A\ell_{i+1} \sin T_i}(C_l \cos T' - E_l \sin T'). \end{aligned}} \tag{5.23}$$

Analytic formula for E_l. Notice that the analytic solution to E_l can be directly derived using a change of variable $\mathrm{d}h = (\frac{\mathrm{d}}{\mathrm{d}t}h)\,\mathrm{d}t$,

$$E_l = \int_0^{T_i} (\frac{\mathrm{d}}{\mathrm{d}t}h) P_l(h)\,\mathrm{d}t = \int_{h(0)}^{h(T_i)} P_l(h)\,\mathrm{d}h = \frac{1}{2l+1}[P_{l+1}(h) - P_{l-1}(h)]\Big|_{c_x}^{c_x \cos T_i + c_y \sin T_i}. \tag{5.24}$$

Here we use the following recurrence relation of the Legendre polynomials: $(2l+1)P_l(h) = \frac{\mathrm{d}}{\mathrm{d}h}(P_{l+1}(h) - P_{l-1}(h))$. [3]

Recurrence formula for C_l. Unlike E_l, it is difficult, if not impossible, to derive a direct representation for C_l. However, our key insight is in reducing the integral expressions to this specific form. Indeed, Wang and Ramamoorthi [102] have developed a recurrence formula for C_l and associated edge integrals,

$$C_l = \frac{1}{l+1}\left[(c_x \sin T_i - c_y \cos T_i)P_l(h(T_i)) + c_y P_l(c_x) + (c_x^2 + c_y^2 - 1)D_l + lB_{l-1}\right], \tag{5.25}$$

where the edge integrals B_l and D_l are given by $B_l = \int_0^{T_i} P_l(h)\,\mathrm{d}t$ and $D_l = \int_0^{T_i} \frac{\mathrm{d}}{\mathrm{d}h}P_l(h)\,\mathrm{d}t$. Their associated recurrence formulae are,

$$B_l = \frac{2l-1}{l}C_{l-1} - \frac{l-1}{l}B_{l-2}, \tag{5.26}$$

$$D_l = (2l-1)B_{l-1} + D_{l-2}. \tag{5.27}$$

[3] The identity still holds for $l = 0$ if we define $P_{-1}(h) \equiv 0$.

The base cases[4] for $l = 0$ are

$$B_0 = T_i,\ D_0 = 0. \tag{5.28}$$

5.5.2 Summary

Based on Equations (5.11) and (5.15), the SH gradient evaluated at one point $\boldsymbol{x}$ can be expressed as

$$\partial_z L_{lm} = \sum_i \sum_j \alpha_{l,j}^m G_{l,j}^{(i)} = \sum_j \alpha_{l,j}^m \underbrace{\left(\sum_i G_{l,j}^{(i)} \right)}_{=: G_{l,j}}. \tag{5.29}$$

We have just derived an analytic formula for $G_{l,j}^{(i)}$ (Equation (5.23)), reducing it to simpler integrals of the Legendre polynomials which are easier to solve.

Analytic SH Coefficients. The edge integrals B_l, C_l and D_l are not only used for evaluating SH gradients, but also building blocks for computing SH coefficients [102]. Therefore, we can simultaneously compute SH coefficients and gradients without much overhead. For completeness, we provide analytic formulae for SH coefficients, which are rewritten and simplified from previous work with respect to our notation.

Similar to Equation (5.29), we can decompose one SH coefficient into the contributions from each individual ZH lobe as $L_{lm} = \sum_j \alpha_{l,j}^m L_{l,j}$. Denoting $H_{l,j}^{(i)}$ as the intermediate quantity for the individual contribution from the j-th ZH lobe and the i-th edge,

$$H_{l,j}^{(i)} = c_z K_l B_l, \tag{5.30}$$

the ZH coefficient $L_{l,j}$ (Equation (5.5)) is given by the following recurrence formula,

$$L_{l,j} = \frac{2l-1}{l(l+1)} \underbrace{\sum_i H_{l-1,j}^{(i)}}_{=: H_{l-1,j}} + \frac{(l-2)(l-1)}{l(l+1)} L_{l-2,j}. \tag{5.31}$$

[4] Additionally, we define $B_{-1} \equiv 0$ and $D_{-1} \equiv 0$.

The base case for $l = 0$ is basically the solid angle subtended by the polygon [3], scaled by K_0,

$$L_{0,j} = K_0 \left[\sum_{i=1}^{N} \arccos\left(\frac{\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i-1}}{\|\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i-1}\|} \cdot \frac{\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i+1}}{\|\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i+1}\|} \right) - (N-2)\pi \right]. \tag{5.32}$$

We also define $L_{-1,j} \equiv 0$ for completeness.

Although previous work and ours both reduce to the same set of edge integrals, the derivation techniques are quite different. Previous work (Equations (5.30) and (5.31)) uses Stokes' Theorem to convert the surface integrals for SH coefficients to the edge integrals. On the other hand, our reduction is based on differentiation of surface integrals under a specific parameterization, followed by term rearrangements with algebraic identities.

5.6 Algorithm

Based on the analytic formulae presented in Section 5.5, we demonstrate a practical algorithm to evaluate SH coefficients and gradients simultaneously, given a shading point and one polygonal light (Section 5.6.1). However, it is still challenging to handle a scene with many area lights, since the method scales linearly in the number of lights.

Fortunately, SH coefficients vary smoothly as the shading point moves. Based on this observation, we develop an efficient algorithm to evaluate the lighting coefficients in the PRT framework, especially when there are multiple uniform polygonal area lights. Our method is outlined in Algorithm 1. We evaluate SH coefficients and gradients for all lights on a sparse grid (Lines 2–8 of Algorithm 1), followed by interpolating SH coefficients of PRT vertices in between (Lines 9–16). In terms of interpolation, we present a gradient-based, tricubic Hermite interpolation method within a 3D grid (Section 5.6.2), which is more accurate than the trilinear interpolation and previous Taylor-series based interpolation [1]. The computation time of interpolation is independent of the number of lights, allowing us to render a scene with hundreds of area lights in real-time.

Algorithm 1. Evaluation of lighting coefficients for every vertex

```
1:  function LightCoeffForPRT
2:      // SH Evaluation on a 3D grid
3:      Build a uniform 3D grid of resolution M^3
4:      for each grid point x do
5:          for each area light in the scene do
6:              Accumulate L_lm(x) and ∇L_lm(x)                    ▷ Algorithm 2
7:          end for
8:      end for
9:      // Gradient-based Interpolation for PRT vertices
10:     for each vertex v do
11:         Find its eight adjacent grid points
12:         for each SH basis (l, m) do
13:             Fetch L_lm and ∇L_lm at the eight grid points
14:             Hermite interpolate L_lm(v)                        ▷ Algorithm 4
15:         end for
16:     end for
17: end function
```

5.6.1 Iterative Evaluation of SH Coefficients and Gradients

We demonstrate the iterative evaluation of *both* SH coefficients and gradients in Algorithm 2. The algorithm takes a shading point $\boldsymbol{x}$, a polygon $\{\boldsymbol{p}_i\}$ with N points, the weights $\{\alpha^m_{l,j}\}$ and central directions $\{\boldsymbol{\omega}_{l,j}\}$ of the precomputed ZH lobes up to degree $l_{\max}$ as input. It outputs $(l_{\max}+1)^2$ SH coefficients $\{L_{lm}\}$ and SH gradients (spatial partial derivatives) $\{\partial_x L_{lm}\}$, $\{\partial_y L_{lm}\}$, $\{\partial_z L_{lm}\}$. [5]

Precomputation. First, we precompute the required quantities for each polygon vertex and edge. Note that the order of polygon vertices does matter. Specifically, the dot product of the face normal $(\boldsymbol{p}_1-\boldsymbol{p}_0)\times(\boldsymbol{p}_2-\boldsymbol{p}_0)$ and $\boldsymbol{p}_0-\boldsymbol{x}$ should be positive. In Lines 2–4 of Algorithm 2, we translate the polygon vertices $\boldsymbol{p}_i$ by $-\boldsymbol{x}$, compute the distances ℓ_i between the vertices and the shading point, and obtain $\boldsymbol{\omega}_i$ by projecting the vertices onto the unit sphere. Then, we build a local frame $(\boldsymbol{\omega}_i, \boldsymbol{\lambda}_i, \boldsymbol{n}_i)$ for each polygon edge (Line 6) and compute its subtended angle T_i (Line 7).

[5] For conciseness, we only show one of the partial derivatives in Algorithm 2. To obtain the other two partial derivatives, we only need to replace n_z by n_x/n_y (lines 15 and 22) and ∂_z by ∂_x/∂_y (lines 33 and 36).

Algorithm 2. SH coefficients and gradients for one polygonal light

function SHCOEFFANDGRAD($\boldsymbol{x}$, N, $\{\boldsymbol{p}_i\}$, $l_{\max}$, $\{\alpha^m_{l,j}\}$, $\{\boldsymbol{\omega}_{l,j}\}$)
  for $i = 0$ to $N-1$ **do** ▷ Precomputation for each vertex
    $\boldsymbol{p}_i = \boldsymbol{p}_i - \boldsymbol{x}$, $\ell_i = \|\boldsymbol{p}_i\|$, $\boldsymbol{\omega}_i = \boldsymbol{p}_i / \ell_i$
  end for
  for $i = 0$ to $N-1$ **do** ▷ Precomputation for each edge
    $\boldsymbol{n}_i = \frac{\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i+1}}{\|\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i+1}\|}$, $\boldsymbol{\lambda}_i = \boldsymbol{n}_i \times \boldsymbol{\omega}_i$
    $T_i = \arccos(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_{i+1})$
  end for
  for $j = 0$ to $2l_{\max}$ **do** ▷ Iterate over $2l_{\max}+1$ ZH lobes
    $\boldsymbol{\omega}_c = \boldsymbol{\omega}_{l_{\max},j}$ ▷ Share ZH lobes, Appendix C.1
    $\{H_{l,j}\} = 0$ ▷ Initialization for ZH coefficients
    $\{G_{l,j}\} = 0$ ▷ Initialization for ZH gradients
    for $i = 0$ to $N-1$ **do** ▷ Iterate over N edges
      $c_x = \boldsymbol{\omega}_c \cdot \boldsymbol{\omega}_i$, $c_y = \boldsymbol{\omega}_c \cdot \boldsymbol{\lambda}_i$, $c_z = \boldsymbol{\omega}_c \cdot \boldsymbol{n}_i$ ▷ Rotation
      $n_z = \boldsymbol{n}_i[z]$ ▷ $\langle \boldsymbol{n}_i, \hat{\boldsymbol{x}} \rangle$
      $A = (c_x^2 + c_y^2)^{1/2}$, $T' = \arctan(c_x / c_y)$ ▷ Equation (5.21)
      $(\{B_l\}, \{C_l\}, \{E_l\}) = \text{RECURRENCE}(c_x, c_y, T_i, l_{\max})$
        ▷ Algorithm 3
      for $l = 0$ to $l_{\max}$ **do**
        $K_l = \sqrt{\frac{2l+1}{4\pi}}$ ▷ SH Normalization factor
        $H_{l,j} \mathrel{+}= c_z K_l B_l$ ▷ Equation (5.30)
        $G_{l,j} \mathrel{+}= \mathrm{G}(n_z, K_l, A, \ell_i, \ell_{i+1}, T_i, T', C_l, E_l)$ ▷ Equation (5.23)
      end for
    end for
    $L_{0,j} = K_0 \times \text{SOLIDANGLE}(N, \{\boldsymbol{\omega}_i\})$ ▷ Base case, Equation (5.32)
    for $l = 1$ to $l_{\max}$ **do**
      $L_{l,j} = \frac{2l-1}{l(l+1)} H_{l-1,j} + \frac{(l-2)(l-1)}{l(l+1)} L_{l-2,j}$ ▷ Equation (5.31)
    end for
  end for
  for $l = 0$ to $l_{\max}$ **do** ▷ ZH factorization
    for $m = -l$ to l **do**
      $L_{lm} = 0$
      $\partial_z L_{lm} = 0$
      for $j \in \{j \,|\, \alpha^m_{l,j} \neq 0\}$ **do** ▷ Sparse weights
        $L_{lm} \mathrel{+}= \alpha^m_{l,j} L_{l,j}$
        $\partial_z L_{lm} \mathrel{+}= \alpha^m_{l,j} G_{l,j}$ ▷ Equation (5.29)
      end for
    end for
  end for
  return $(\{L_{lm}\}, \{\partial_z L_{lm}\})$
end function

Algorithm 3. Iterative evaluation of edge integral recurrences

1: **function** RECURRENCE(c_x, c_y, T_i, $l_{\max}$)
2: $B_0 = T_i$, $C_0 = c_x \sin T_i - c_y \cos T_i + c_y$ ▷ Base cases
3: $D_0 = 0$, $E_0 = c_x \cos T_i + c_y \sin T_i - c_x$
4: **for** $l = 1$ to $l_{\max}$ **do**
5: $B_l = \frac{2l-1}{l} C_{l-1} + \frac{l-1}{l} B_{l-2}$ ▷ Equation (5.26)
6: $D_l = (2l-1) B_{l-1} + D_{l-2}$ ▷ Equation (5.27)
7: $C_l = \mathrm{C}(c_x, c_y, T_i, l, D_l, B_{l-1})$ ▷ Equation (5.25)
8: $E_l = \mathrm{E}(c_x, c_y, T_i, l)$ ▷ Equation (5.24)
9: **end for**
10: **return** $(\{B_l\}, \{C_l\}, \{E_l\})$
11: **end function**

Evaluation of individual ZH coefficients and gradients. Starting from Line 9 of Algorithm 2, we evaluate the ZH coefficients (indicated in the orange background) and gradients (indicated in the purple background) for up to $(2l_{\max}+1)$ ZH lobes. We use $\boldsymbol{\omega}_c$ to indicate the central direction of the j-th ZH lobe (Line 10), given the lobe sharing strategy in Appendix C.1. The contributions to ZH coefficients and gradients will be accumulated for every polygon edge (Lines 13–24).

For each edge, the local coordinates (c_x, c_y, c_z) of $\boldsymbol{\omega}_c$ in the edge's local frame are calculated in Line 14. We calculate the values n_z,A and T' in Lines 15 and 16, which are required for the gradient evaluation. The edge integrals B_l, C_l and E_l are evaluated iteratively (Line 17) from $l = 0$ to $l_{\max}$. We demonstrate the computation of the recurrence formulae (Equations (5.24)–(5.27)) in Algorithm 3. For each band l, the values $H_{l,j}$ (Line 21) and the ZH gradients $G_{l,j}$ (Line 22) are updated based on Equations (5.30) and (5.23) respectively.

Finally, the ZH coefficients $L_{l,j}$ are evaluated in Lines 25–28, based on another recurrence formula related to $H_{l,j}$ (Equations (5.31) and (5.32)).

ZH factorization. The final step is to reconstruct SH coefficients and gradients from the evaluated ZH coefficients $L_{l,j}$ and gradients $G_{l,j}$ (Lines 30–39). The ZH factorization weights $\alpha_{l,j}^m$ are precomputed according to the work by Nowrouzezahrai et al. [75]. The sparsity of

Algorithm 4. Tricubic Hermite interpolation from function values f_i and gradients $\nabla f_i = (\partial_x f_i, \partial_y f_i, \partial_z f_i)$ of the eight adjacent grid points $i = 0, 1, \ldots, 7$

function TRICUBICHERMITE($\boldsymbol{p}$, *gridSize*, $\{f_i\}$, $\{\nabla f_i\}$)
  $(x_R, y_R, z_R) = gridSize$ ▷ Size of a grid voxel
  $(x, y, z) = \boldsymbol{p}$ ▷ Vertex coordinates inside the grid voxel
  for $i = 0$ to 3 **do** ▷ Interpolate along the x-axis
    $g_i = \text{HERMITE1D}(x, \{0, f_{2i}, \partial_x f_{2i}\}, \{x_R, f_{2i+1}, \partial_x f_{2i+1}\})$
    Interpolate ∇g_i linearly based on ∇f_{2i} and ∇f_{2i+1}
  end for
  for $i = 0$ to 1 **do** ▷ Interpolate along the y-axis
    $h_i = \text{HERMITE1D}(y, \{0, g_{2i}, \partial_y g_{2i}\}, \{y_R, g_{2i+1}, \partial_y g_{2i+1}\})$
    Interpolate ∇h_i linearly based on ∇g_{2i} and ∇g_{2i+1}
  end for
  $q(x, y, z) = \text{HERMITE1D}(z, \{0, h_0, \partial_z h_0\}, \{z_R, h_1, \partial_z h_1\})$
  return $q(x, y, z)$ ▷ Interpolate along the z-axis
end function
function HERMITE1D(x, $\{x_L, f_L, f'_L\}$, $\{x_R, f_R, f'_R\}$)
  $x_\Delta = x_R - x_L$, $s = (f_R - f_L)/x_\Delta$
  $a = f_L$, $b = f'_L$, $c = \frac{s - f'_L}{x_\Delta}$, $d = \frac{f'_L + f'_R - 2s}{x_\Delta^2}$ ▷ Equation (5.35)
  return $a + b(x - x_L) + c(x - x_L)^2 + d(x - x_L)^2(x - x_R)$ ▷ Equation (5.33)
end function

weights is maximized, so the SH reconstruction is efficient.

Summary. Evaluating ZH coefficients and gradients (Lines 9–29 of Algorithm 2) takes $O(Nl_{\max}^2)$ time and reconstructing SH values with the ZH factorization (Lines 30–39) takes $O(l_{\max}^3)$ time. Note that the last ZH factorization step is quite fast, since the weights $\alpha_{l,j}^m$ are sparse. The overall storage required is $O(l_{\max}^2)$. Both the time and space complexity are the same as in previous work [102]. In terms of implementation, computing SH gradients along with SH coefficients only requires minimal effort (lines in Algorithm 2 with the purple background).

5.6.2 Gradient-Based Interpolation

Given SH coefficients and gradients evaluated on a 3D grid, we can interpolate for any inside point (vertex in PRT) according to its eight adjacent grid points. Note that the interpolation time only depends on the highest SH degree $l_{\max}$ and is independent of the number of lights,

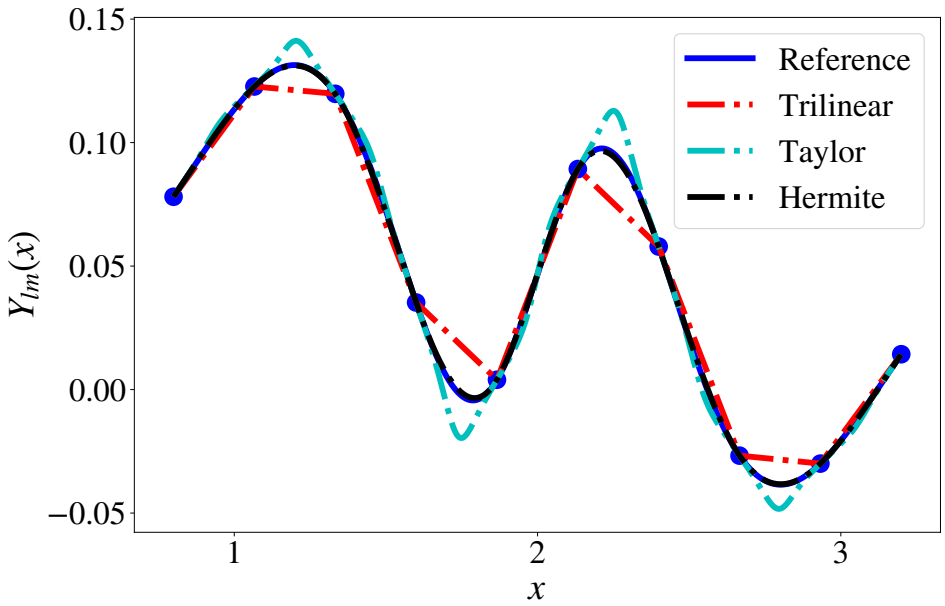


Figure 5.4. Given a rectangular area light and a shading point, we plot the SH coefficients for $(l,m) = (7,2)$ as the shading point moves along the x-axis. The reference curve (blue solid line) is densely sampled at 2000 points. Alternatively, we evaluate SH coefficients and gradients at 10 points (blue dots) and interpolate the coefficients in between. The trilinear interpolation (red dashed line) and the Taylor-series based interpolation (cyan dashed line) [1] result in insufficient accuracy, while the cubic Hermite interpolation result (black dashed line) matches the reference almost perfectly.

making our method scalable to many lights.

In terms of interpolation methods, one can interpolate SH coefficients trilinearly, without using SH gradients at all. Previous work [1] uses an interpolation method based on Taylor series. They approximate the coefficients by the first-order Taylor polynomial from each adjacent grid point, then combine the results using the inverse distance weighting [103]. Both interpolation methods result in limited accuracy (see Figures 5.4 and 5.7). Instead, we use a more principled Hermite interpolation [98], approximating the interpolant as a tricubic polynomial. We first provide details of the cubic Hermite interpolation in 1D. Then, we discuss its extension to 3D.

1D cubic Hermite interpolation. Suppose the function we are going to interpolate $f(x)$ is continuous on $[x_L, x_R]$ and we know its values f_L, f_R and the first-order derivatives f'_L, f'_R at the endpoints, respectively. The cubic Hermite interpolant $q(x)$ is a cubic polynomial with four

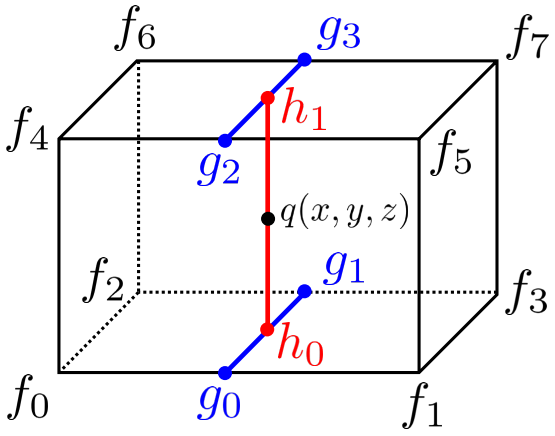


Figure 5.5. Illustration of 3D Hermite interpolation.

unknown coefficients a, b, c and d [98],

$$q(x) = a + b(x - x_L) + c(x - x_L)^2 + d(x - x_L)^2(x - x_R), \tag{5.33}$$

satisfying

$$q(x_L) = f_L,\ q(x_R) = f_R,\ q'(x_L) = f'_L,\ q'(x_R) = f'_R. \tag{5.34}$$

We can obtain the four unknown coefficients by solving this linear system (Equation (5.34)),

$$a = f_L,\ b = f'_L,\ c = (s - f'_L)/x_\Delta,\ d = (f'_L + f'_R - 2s)/x_\Delta^2, \tag{5.35}$$

where $x_\Delta = x_R - x_L$ and $s = (f_R - f_L)/x_\Delta$. We demonstrate it in Lines 15–19 of Algorithm 4. The 1D curve plot in Figure 5.4 indicates the accuracy benefit comparing to other interpolation methods.

3D tricubic Hermite interpolation. Tricubic Hermite interpolation can be done by performing the 1D cubic Hermite interpolation along the three axes progressively (see Figure 5.5). For any intermediate points, the SH coefficients are Hermite interpolated but we need to know the SH gradients (first-order derivatives) as well. In theory, applying Hermite interpolation to

Table 5.1. Scene configurations of all results.

Scene		Triangles	Lights
Dragon & Bunny	Figure 5.1	1.28M	723
Monkey	Figure 5.7	71.2K	118
Plants	Figure 5.8	190K	2
Asian Dragon	Figure 5.9	1.42M	181
Room	Figure 5.10	1.71M	344
Buddha	Figure 5.11	422K	210

the first-order derivatives requires the second-order derivatives, which will involve extra cost to evaluate. For efficiency, we interpolate SH gradients trilinearly and get satisfactory results.

We provide the pseudocode of the 3D tricubic Hermite interpolation in Algorithm 4. For a point $\boldsymbol{p}$ in a grid voxel with size (x_R, y_R, z_R), we translate it so that the bottom-left corner is at $(0,0,0)$. First, we interpolate along the x-axis (Lines 4–7), calculating the function values $g_0, \ldots, g_3$ by the 1D Hermite interpolation and gradients $\nabla g_0, \ldots, \nabla g_3$ by the linear interpolation (see the blue points in Figure 5.5). We then interpolate along the y-axis (Lines 8–11) and obtain the values h_0, h_1 and $\nabla h_0, \nabla h_1$ (see the red points in Figure 5.5). Finally, we compute the interpolant value $q(x,y,z)$ by interpolating along the z-axis (Line 12).

Note that theoretically, the interpolation result depends on the order of the individual 1D steps. However in practice, we do not observe significant differences when we change the order.

5.7 Results

We implement Algorithms 1–4 in GPU shaders and compute SH lighting coefficients for each vertex in a scene. The lighting coefficients are used in a PRT system, which is implemented within the Falcor open-source real-time rendering framework [6]. We run our algorithm on a few scenes with multiple polygonal area lights using an NVIDIA RTX 2080 Ti GPU. The scene configurations and performance statistics are summarized in Tables 5.1 and 5.2, respectively. We release our code and data in the supplementary material of the original publication.

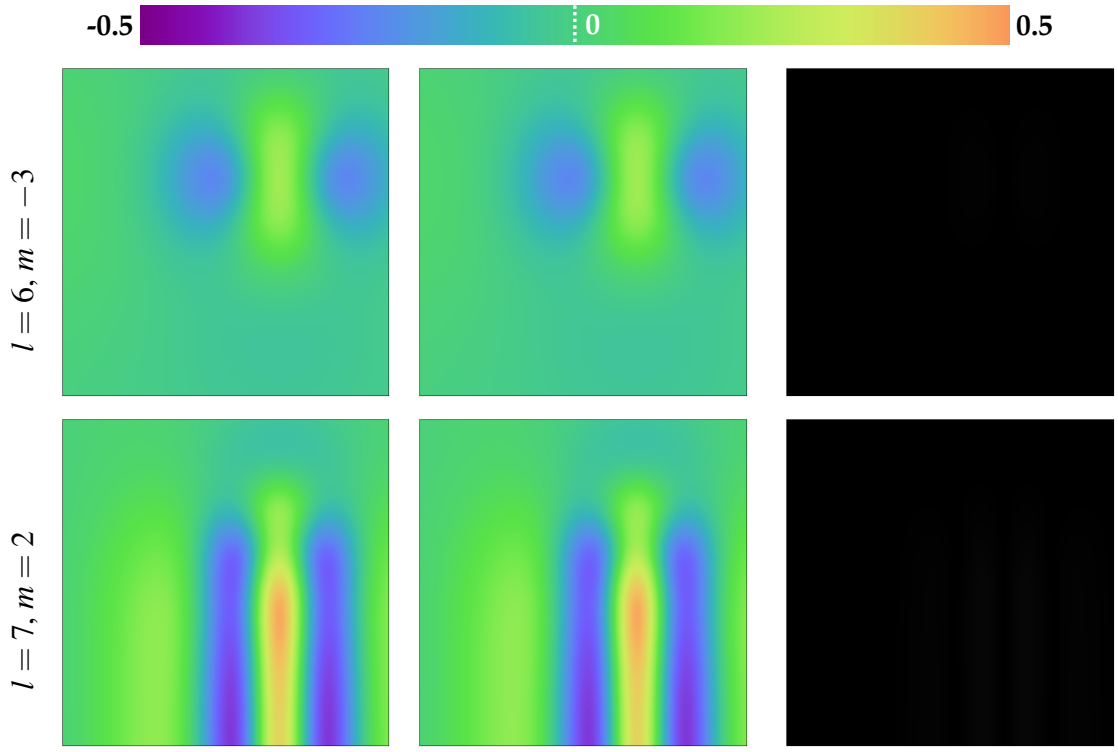
Table 5.2. Performance statistics of all results. We compare running time and image mean absolute errors (MAE) to those by Wang and Ramamoorthi [102], which we treat as the Reference. Note that the total running time of our method also includes other necessary operations such as rasterization. We also provide the running time of evaluating *only* SH coefficients to highlight the low overhead of SH gradient evaluation.

Scene	Grid Reso.	Eval. coeff.	Coeff. & grad.	Interp.	**Total**	FPS	MAE ($\times 10^{-3}$)	Ref.	Speed up
Dragon & Bunny	8^3	16.5	20.2	6.6	**27.9**	35.8	0.18	110K	3943×
Monkey	8^3	1.3	1.8	0.4	**3.3**	303	0.29	177	35×
Plants	8^3	0.08	0.1	2.1	**3.1**	323	0.17	9.4	3×
Asian Dragon	8^3	2.3	3.0	8.1	**12.7**	78.7	1.35	3.22K	254×
Room	8^3	4.3	5.8	10.3	**17.7**	56.5	0.34	6.85K	387×
Buddha	8^3	4.8	5.7	6.0	**12.7**	78.7	0.07	28.6K	2252×

5.7.1 Validation and Evaluation

Validation of Analytic SH Gradients. To validate our derivation of analytic SH gradients, we compare SH gradients evaluated using our method (Algorithm 2) and finite differences (FD) based on the work by Wang and Ramamoorthi [102]. Given a rectangular area light with the bottom-left corner at $(-5,-5,1)$ and the top-right corner at $(5,5,1)$, the ∂_x-component of SH gradients is evaluated in another square region whose bottom-left corner is at $(3,3,0)$ and top-right corner is at $(6,6,0)$. We visualize the derivatives as 2D false-colored images in Figure 5.6. Our analytic formulae agree with the numerical FD results, except for some negligible differences caused by the FD step size δ (we choose $\delta = 10^{-3}$). Note that computing derivatives with the central FD requires SH coefficient evaluation at multiple points (two for each axis), while our SH gradient evaluation can come along with a single SH coefficient evaluation, causing minimal overhead. We compare the performance numbers of these two methods with a CPU-based C++ implementation; our method is 3× faster than FD.

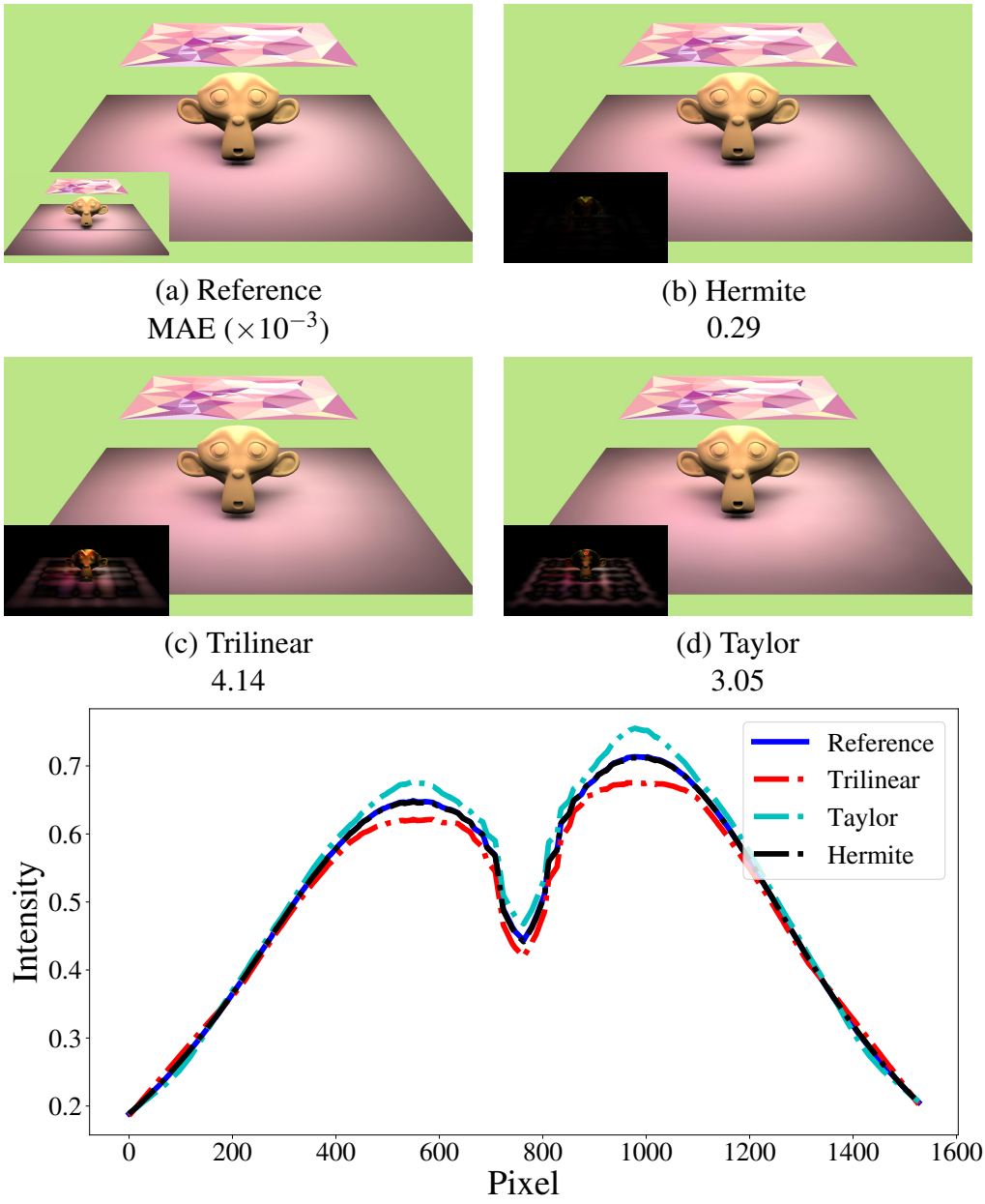
Interpolation Methods. We have already demonstrated in Figure 5.4 that the Hermite interpolation is most accurate in 1D cases. In terms of its 3D extension, we compare renderings using different interpolation methods in Figure 5.7. Hermite interpolation results in an order of



(a) FD (b) Ours (c) 20× Abs. diff.

Figure 5.6. We show visualization plots of SH gradients (∂_x-component) for $(l,m) = (6,-3)$ in the top row and $(l,m) = (7,2)$ in the bottom row. Images in column (a) are computed using finite differences (FD) based on the work by Wang and Ramamoorthi [102], and images in column (b) are computed with our method (Algorithm 2). In these images, each pixel stores a partial derivative value encoded in false colors; The (20×) absolute differences between FD results and our results are given in column (c), indicating the correctness of our derivation and algorithm.

magnitude smaller error than that of trilinear interpolation and Taylor-series based interpolation. Note that Taylor-series based interpolation [1] requires SH gradients, thus can also benefit from our analytic SH gradient evaluation. Although the mean absolute error (MAE) numbers are relatively small, there are regions with significant inaccuracies in other methods. Checking the error maps in Figure 5.7(b–d), the result using Hermite interpolation has significantly fewer pixels with large differences from the reference. Further, we check and plot the image intensity values along a scan line in Figure 5.7(e). The black curve representing Hermite interpolation is almost identical to the reference blue curve, while the red and cyan curves, representing trilinear and Taylor-series based interpolation respectively, deviate from the reference.



(a) Reference
MAE ($\times 10^{-3}$)

(b) Hermite
0.29

(c) Trilinear
4.14

(d) Taylor
3.05

(e) Scan line intensity plot

Figure 5.7. Accuracy comparison of different interpolation methods: (b) tricubic Hermite interpolation, (c) trilinear interpolation, and (d) Taylor-series based interpolation [1]. The reference image (a) is rendered by computing the lighting SH coefficients at every vertex. We achieve almost the same image quality (b) by computing SH coefficients and gradients in a 3D grid with resolution 8^3, and Hermite interpolating the light coefficients for each vertex. The ($10\times$) absolute error images are given in the bottom-left insets, as well as the corresponding mean absolute error (MAE) numbers. We also plot image intensity curves (e) for different interpolation methods along a scan line (illustrated in the inset of (a)).

Scalability with Multiple lights. In Figure 5.8(a), we compare the total lighting coefficient computation time for all vertices (time for other stages in the rendering pipeline such as rasterization is excluded) with the previous method [102]. The running time of the previous method goes up linearly with the number of lights as expected, while our method has only a small time cost even with 512 lights. This is because the expensive computation for every light is done on a sparse grid in our method. Our technique can render a scene with hundreds of area lights in real-time, which was not previously possible. We further analyze the performance of our method and plot the running time of each step in Figure 5.8(b). The SH coefficient and gradient evaluation time scales linearly with the increasing number of lights, but the performance impact is mitigated since we use a sparse grid. The Hermite interpolation for each vertex requires essentially constant time.

Grid Resolution. We demonstrate how the grid resolution influences the rendering performance and accuracy in Figure 5.9. Even though the result with resolution 4^3 already yields good visual quality (inspection of error images shows subtle differences in shading on the dragon and floor), the image accuracy improves as the grid resolution becomes higher. But using a finer grid also requires longer computation time and larger storage overhead. To balance performance and accuracy, we use a resolution of 8^3 in all the results.

Relation to Source Radiance Fields. In previous work [122], source radiance fields (SRF) are precomputed and stored for efficient incident radiance evaluation. To support dynamic lighting, a 5D SRF is required for each area light, causing additional storage overhead that scales linearly in the number of lights. However, our method only stores SH coefficients and gradients at sparse grid points, which is independent of the number of lights. Moreover, the source radiance fields at intermediate points are interpolated, which can also benefit from our gradient-based interpolation.

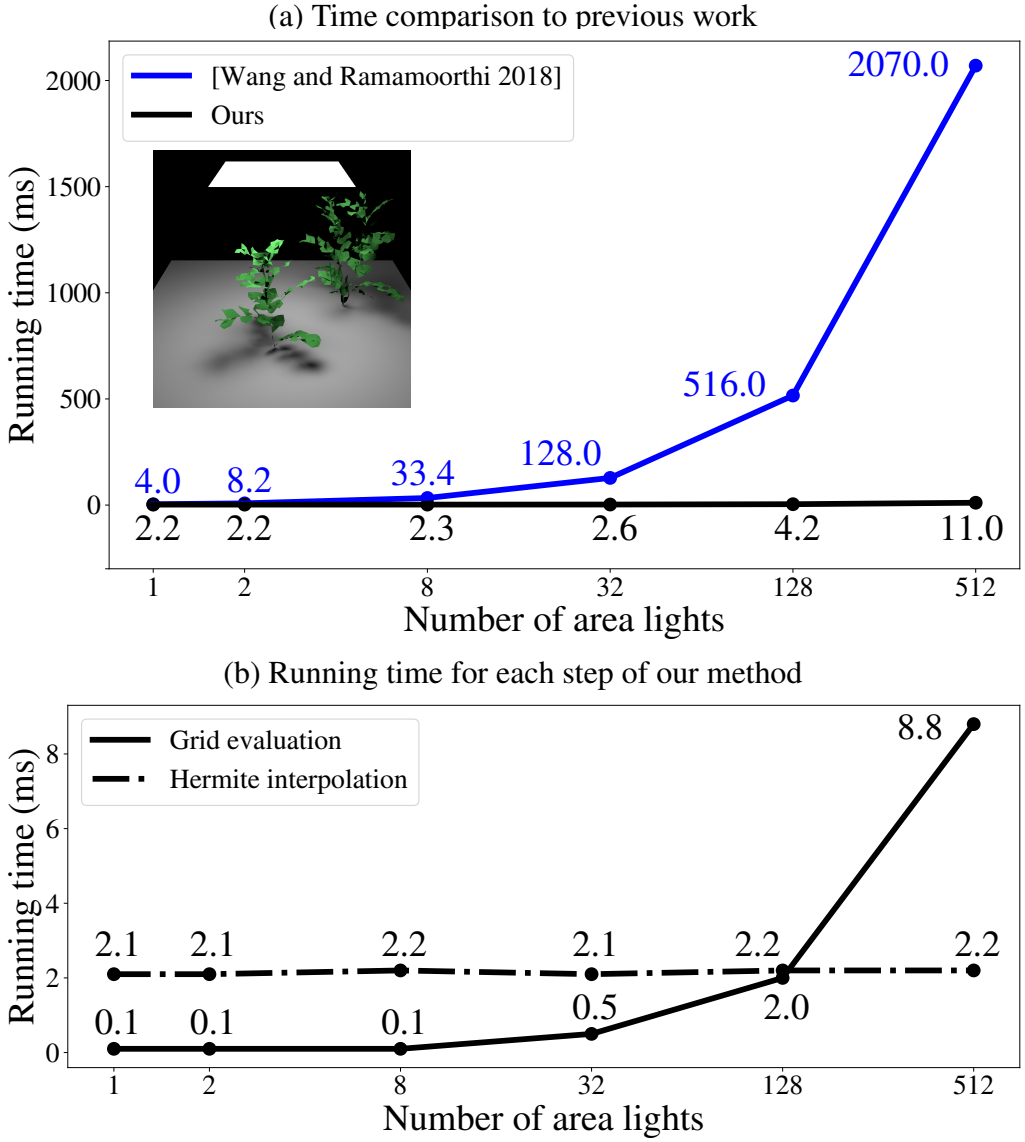
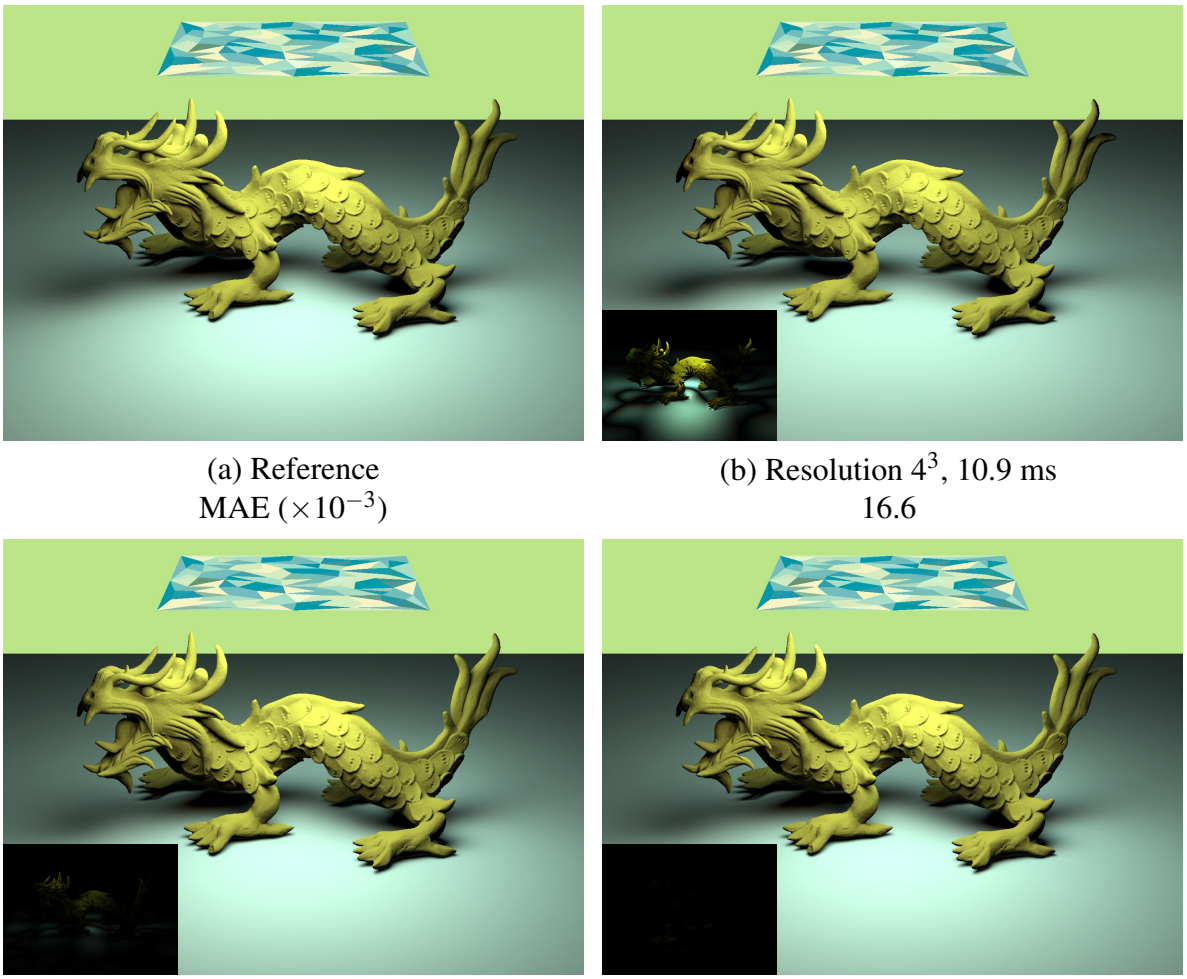


Figure 5.8. Plots of SH coefficient computation time at all vertices with increasing numbers of area lights. The scene is shown in the inset of (a); we gradually subdivide the rectangular area light up to 512 triangles. (a) Previous work scales linearly in the number of area lights, so they cannot handle many lights in real-time. On the other hand, our method is insensitive to the increase in the number of lights, since we only need to compute the light coefficients and gradients on a sparse grid. (b) Breaking down the running time of our method, we can see the time for grid evaluation is also linear in the number of lights (but still efficient because of the sparse grid), while Hermite interpolation costs essentially constant time.



(a) Reference
MAE ($\times 10^{-3}$)

(b) Resolution 4^3, 10.9 ms
16.6

(c) Resolution 8^3, 12.1 ms
1.35

(d) Resolution 16^3, 34.4 ms
0.15

Figure 5.9. Performance and accuracy comparison with increasing resolutions of 3D grids. The bottom-left insets show ($5\times$) absolute error images.

5.7.2 Main Results

We now present additional results of rendering more complex scenes. Please see the supplementary video in the original publication for animated versions of Figures 5.1, 5.10, and 5.11, rendered at real-time frame rates (35–80 fps)

Textured Lights. Given a textured area light, we break the light source into smaller polygons (triangles) that are each uniformly emissive. We show an example in Figure 5.10, in which a room is illuminated by a blue and a pink textured light (see the bottom-left inset of

Figure 5.10. A living room illuminated by two textured lights. Light sources and scene layout are illustrated in the inset figure.

Figure 5.10). Compared to the previous method [102], we can render this scene with hundreds of lights (and 1.7M polygons) in real-time, achieving a more than two orders of magnitude speed up.

Glossy Reflection. We also show two examples with glossy materials in Figures 5.1 and 5.11. We compute the glossy reflection by extending Equation (5.1) to a triple product SH integral of lighting, BRDF and precomputed cosine-weighted visibility [72, 89]. Since the time complexity of the triple product integral computation is $O(l_{\max}^5)$ [72], we bandlimit the lighting and visibility SH coefficients with $l_{\max} = 4$. Phong BRDFs are used in all these examples, represented by SH coefficients with $l_{\max} = 8$. Our method focuses on evaluating the lighting SH coefficients only, and is orthogonal to the glossy PRT framework.

In Figure 5.1, we show images rendered with three textured lights of gradually changing colors. These textured lights are made up of more than seven hundred uniform triangular light sources in total, and each of them is allowed to move independently, ultimately forming a

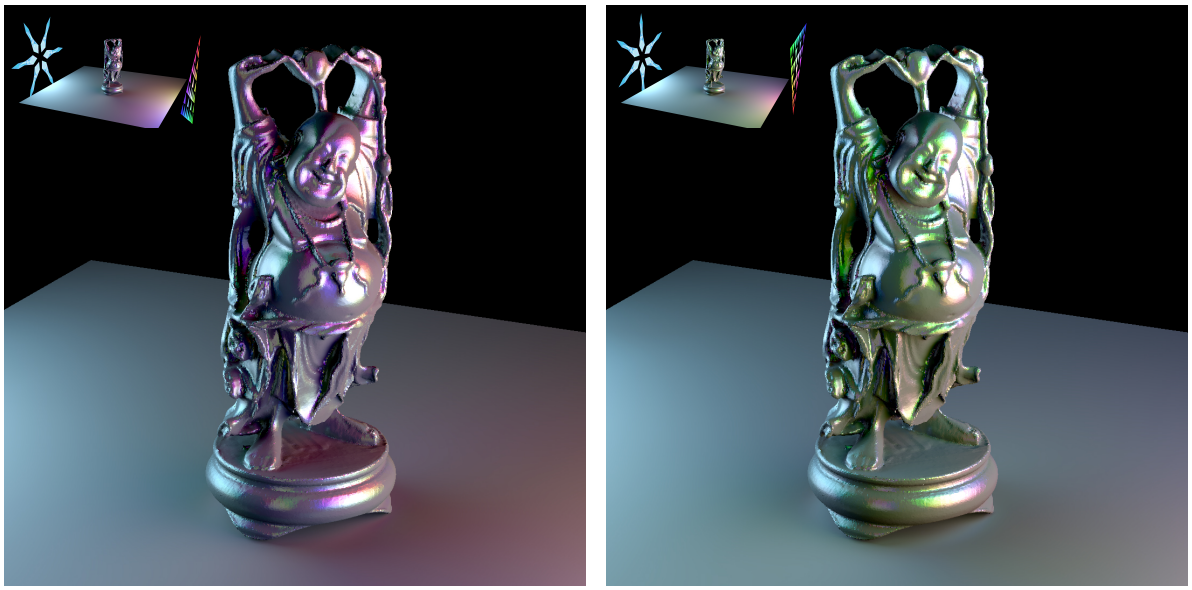


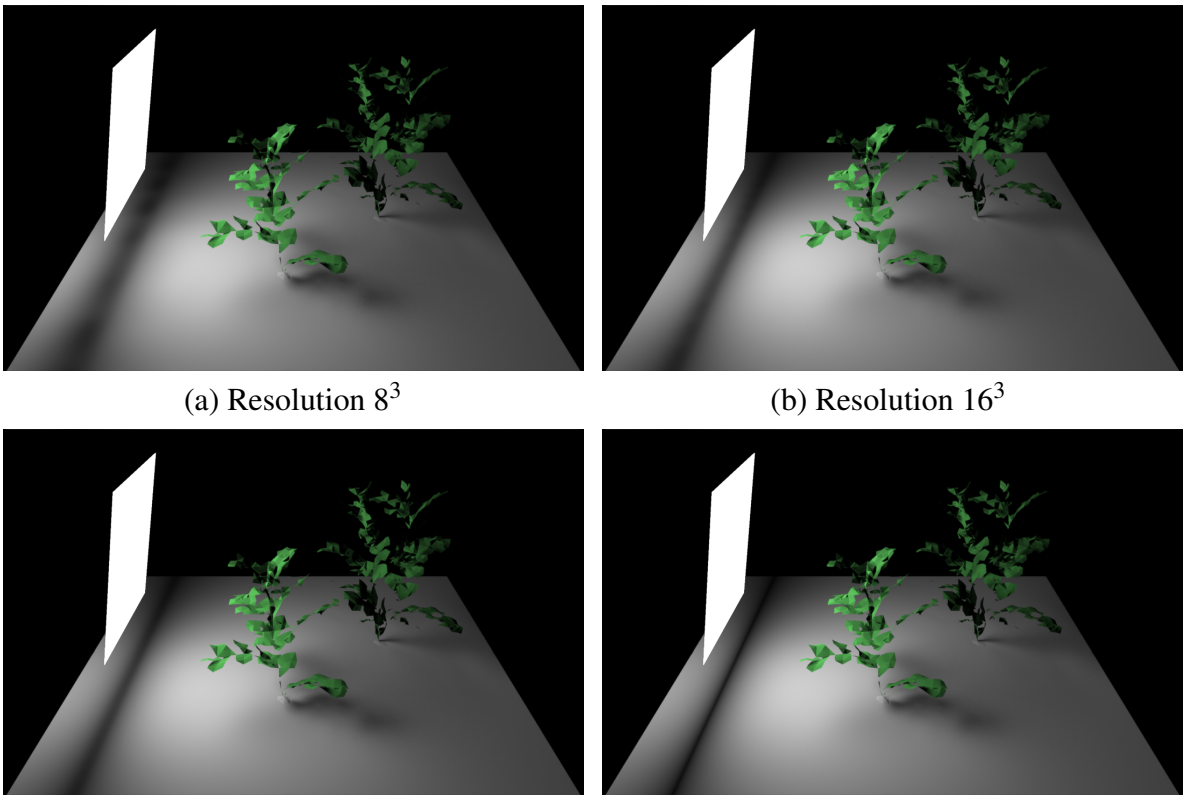
Figure 5.11. Glossy reflections caused by more complex light sources. Light sources and scene layout are illustrated in the inset figure.

pattern which is not even strictly a textured light source (Figure 5.1(right)). Colors of the glossy highlights change significantly as we transform the lights. Even though there are hundreds of independent dynamic lights, we are still able to render this scene in real-time, which might be challenging for source radiance fields [122], since it requires precomputation for each light source.

In Figure 5.11, we illuminate a Buddha model on the ground with two lights of irregular shapes (see the insets). The left light is a blue snow flake and the right one is a colorful fractal triangle. As we rotate the lights, the glossy highlights on the buddha change from purple to green. The highlights on the ground also vary according to the light transformation.

5.7.3 Limitations and Future Work

On regions close to the light source or at grazing angles, there can be high-frequency lighting variations that require a fine grid for accurate SH interpolation. Figure 5.12 shows a scene with a double-sided area light inside its bounding box. The shading that is on the ground and close to the light's grazing angle looks blurry when the grid resolution is 8^3. We will



(a) Resolution 8^3 (b) Resolution 16^3

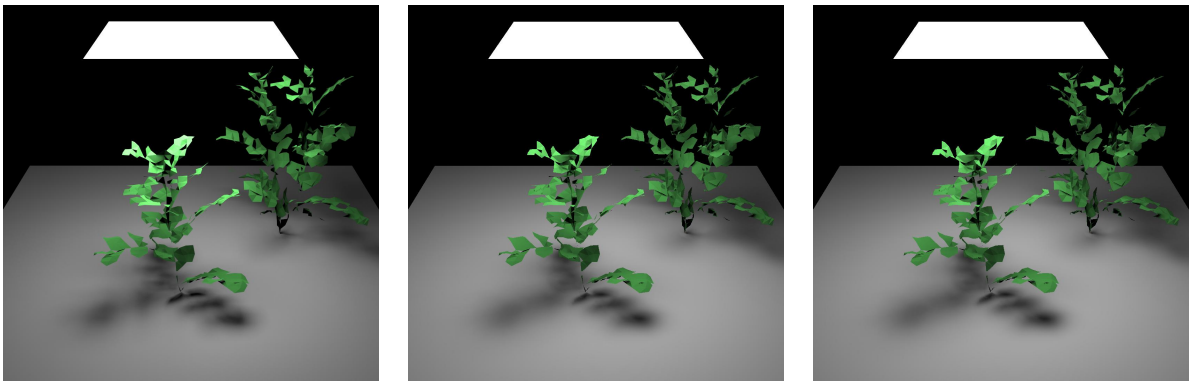
(c) Resolution 32^3 (d) Reference

Figure 5.12. An example with a double-sided area light source inside the scene. Due to the high-frequency light variations, we need finer grids for accurate SH interpolation.

have more accurate interpolation results given grids with higher resolutions. Using multi-level adaptive grids [23] may further improve the performance and accuracy in such cases.

Our method is based on PRT using spherical harmonics, which approximates path tracing and captures only low-frequency effects due to the band-limited SH. In Figure 5.13, we compare our result to the image rendered using path tracing. There are some subtle differences at the shadows on the ground and the plant leaves near the light source. Nevertheless, we match the result generated by Wang and Ramamoorthi [102] almost perfectly, which also uses PRT. How to improve the accuracy of all-frequency effects using spherical harmonic PRT is orthogonal to our work and beyond the scope of this work. We also hope our work can inspire future research on real-time path tracing.

In this work, we focus on accurate analytic formulae for SH gradients. In certain



(a) Path tracing ≈ (b) PRT (Ours) = (c) PRT (Wang and Ramamoorthi 2018)

Figure 5.13. Comparison against path tracing. Our result (b) is close to the image rendered using path tracing (a) in spite of subtle differences at the shadows on the ground and the plant leaves near the light source, while it matches the result of Wang and Ramamoorthi [102] almost perfectly.

cases, e.g., the light source is far away, one sample per pixel for numerical integration could be sufficient and faster than analytic evaluation. Switching between numerical and analytic evaluations according to some heuristics [115] might be potentially helpful. In addition, applying analytical approximations [58] might make the gradient evaluations more efficient.

Note that non-uniform or textured lights can be handled by our method, simply by breaking the light source into smaller uniform components. With techniques demonstrated in prior analytic methods [3, 10, 11], it might be possible to extend our analytic SH gradients for piecewise linear area light sources. Moreover, we are currently limited to uniform angular emission rather than, for example, spotlights. We seek to lift this limitation in the future, perhaps by developing fast boundary numerical integration schemes.

Finally, note that we do not currently address multiple lights shadowing each other (although PRT methods that support dynamic shadows can partially address this situation). We believe the boundary integral formulation of this work could also generalize occluded irradiance gradients [2] to SH gradients by incorporating polygon depth clipping.

5.8 Conclusion

We have presented a novel technique of prefiltering multiple area lights, which enables scaling PRT to hundreds of independent area lights in real-time. Our prefiltering method is based on a novel analytic derivation for SH gradients from uniform polygonal area lights, showing how to reduce the calculations to a boundary integral and ultimately to an earlier recurrence for SH coefficients. The derivation fills an important gap in both SH and PRT methods, as well as more recent differential rendering techniques. While the derivation is complicated, the actual implementation is simple, requiring only a few additional lines of code beyond those for SH coefficients. We show how gradients can be used for Hermite interpolation with very high accuracy and sparse grid sizes. Crucially, we can accumulate the contributions of hundreds of lights with only minor overhead.

Prefiltering many area lights in PRT represents only one possible application of SH gradients. They could also be used for importance sampling the radiance field from multiple area lights in offline rendering, and for extensions such as path guiding. We simply need to sample the Hermite-interpolated SH lighting at each shading point or pixel. Given that SH gradients are a fundamental mathematical quantity, we believe there are many other interesting possibilities in rendering and beyond.

5.9 Acknowledgements

This chapter is based on the material to appear in ACM Transactions on Graphics, 2020 ("Analytic Spherical Harmonic Gradients for Real-Time Rendering with Many Polygonal Area Lights", Lifan Wu, Guangyan Cai, Shuang Zhao, and Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 6

Conclusion and Future Work

We have presented three different *appearance preserving prefiltering* techniques for efficient rendering of volumetric scattering models, displacement-mapped surfaces, and multiple polygonal area lights, respectively. Our first work exploits an optimization based approach to compute downsampled scattering parameters. It requires gradient image estimation and iterative optimization, which takes tens of CPU core hours. Our second work uses the effective BRDFs as a proxy indicator of appearance matching and allows computing the prefiltered models directly without expensive iterative optimizations. The key component of our prefiltered models is the novel scaling function that captures the change of shadowing-masking and interreflection effects caused by the downsampling of micro-geometries. Our last work leverages analytic formulae for spherical harmonic lighting coefficients and gradients, enabling prefiltering the contributions of hundreds of area lights at real-time frame rates. Although these three approaches tackle the challenge of prefiltering for rendering complex scenes in different ways, our prefiltered models offer a compact scene representation and are capable of preserving the original appearance accurately.

We have demonstrated not only the theoretical analysis of prefiltering but also practical algorithms. This dissertation has taken one step forward towards general and efficient appearance preserving prefiltering methods for rendering complex scenes. We believe our work can inspire future research in this direction and benefit many other applications in photorealistic rendering

and beyond. Here, we discuss the following possible future directions:

Level-of-detail rendering. Level-of-detail (LoD) rendering is necessary for large scenes such as cities and forests, or for performance-critical applications such as video games and virtual reality. In Chapters 3 and 4, we have presented promising LoD rendering results of objects represented by regular 3D volumetric scattering parameters or 2D displacement maps. However in general, scene objects are usually described by triangular meshes. Compared to regular 2D or 3D grids, the mesh representation is irregular and non-uniform, thus causes extra difficulty for prefiltering. It is even harder to preserve the appearance for meshes associated with texture maps. Finding a continuous and consistent (i.e., not showing popping artifacts when switching between two consecutive mipmap levels) prefiltering method for general mesh representation remains very challenging.

Multi-scale appearance modeling. Micro-appearance models can provide detailed geometric and scattering configuration and reproduce accurate appearance at all scales. The high-quality realism comes at the cost of enormous data size and long computation time. In contrast, microfacet models describe surface geometry in a statistical manner, which are compact and efficient to use. But they are only capable of producing appearance at the macroscopic scale, lacking spatial variety. The prefiltering methods in Chapters 3 and 4 take the first step to blend the advantages of these two types of appearance models. Our prefiltered models can accurately resemble the appearance at intermediate levels. One major drawback is that our prefiltering is material-dependent, requiring precomputation for each individual material. Possible extensions include using a material-independent data-driven approach, or designing a multi-scale statistical surface reflectance model with better generality. In addition, it is necessary to consider wave effects when the size of geometric details is close to the wavelength of light.

Inverse rendering. Inverse rendering provides a bridge across computer graphics and computer vision. It is a principled approach to solve many problems in visual computing such as appearance matching and 3D reconstruction. Prefiltering is one application of inverse rendering:

given a reference image, we aim to find the optimal prefiltered model that closely matches the input appearance. This inverse problem is ill-conditioned in general, since there are a large number of unknown variables and potential local minima involved. Without a careful design of scene and optimization configurations, it is difficult to converge. In Chapter 3, we regularize the inverse problem by reducing the number of unknown variables using voxel clustering, and set optimization hyperparameters manually. It is worth further exploring the theoretical properties of this inverse problem and practical algorithms that can accelerate convergence.

Differentiable rendering. A major challenge in inverse rendering is computing derivative with respect to arbitrary scene parameters. In Chapter 5, we have demonstrated analytic SH gradients and their use in real-time PRT. Higher-order derivatives of SH coefficients in analytic form may further improve the accuracy. Although in general, it is intractable to derive analytic derivatives for light transport due to the high complexity. Differentiable rendering using Monte Carlo estimators, which we have used in Chapter 3, focuses on derivatives with respect to scattering parameters. Several recent techniques [61, 65, 117] address derivatives with respect to geometric parameters, which are more difficult to solve due to the discontinuities at the boundaries. Though the estimation of general scene derivatives has so far been computationally expensive, it can become more practical with better sampling strategies and variance reduction techniques [116]. Since the estimated gradient images are usually noisy due to the limited sample budget, it may be useful to apply state-of-the-art denoising methods [20, 100] as a post process. Prefiltering approaches based on inverse rendering will also benefit from from high-performance differentiable rendering systems [35, 36, 73].

Appendix A

Appendix for Chapter 3

A.1 Rendering Gradient Images

Under the path tracing framework, the intensity $\tilde{I}_r(p)$ of pixel p in image $\tilde{I}_r$ is estimated by a path integral

$$\tilde{I}_r(p) = \int_{\Omega} f(\bar{x})\,\mathrm{d}\bar{x}, \tag{A.1}$$

where Ω is the space containing all light paths connecting pixel p and a light source and $f(\bar{x})$ denotes the contribution of light path $\bar{x}$. Let $\bar{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n+1})$ with segment $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ intersecting pixel p on the virtual sensor and $\boldsymbol{x}_{n+1}$ lying on a light source. For each $1 \leq v \leq n$, let i_v denote the index of the downsampled voxel containing point $\boldsymbol{x}_v$. It holds that

$$f(\bar{x}) = \left(\prod_{v=1}^{n} T_v \, \bar{\sigma}_{t,i_v} \, F_v \right) L_e(\boldsymbol{x}_{n+1}, \boldsymbol{x}_n), \tag{A.2}$$

where L_e denotes the attenuated incoming radiance, T_v is the transmittance between $\boldsymbol{x}_{v-1}$ and $\boldsymbol{x}_v$, $\bar{\sigma}_{t,i_v}$ is the downsampled density at voxel i_v, and F_v is the scaled phase function at $\boldsymbol{x}_v$ evaluated with incoming and outgoing directions respectively given by $\boldsymbol{x}_{v-1}$ and $\boldsymbol{x}_{v+1}$:

$$F_v := \hat{f}_{i_v} \left(\frac{\boldsymbol{x}_v - \boldsymbol{x}_{v-1}}{\|\boldsymbol{x}_v - \boldsymbol{x}_{v-1}\|} \to \frac{\boldsymbol{x}_{v+1} - \boldsymbol{x}_v}{\|\boldsymbol{x}_{v+1} - \boldsymbol{x}_v\|} \right). \tag{A.3}$$

Let w be a weight factor for some voxel cluster and lobe. Then,

$$\tilde{I}'_r(p) := \frac{\partial}{\partial w}\tilde{I}_r(p) = \int_\Omega \frac{\partial f}{\partial w}(\bar{x})\,\mathrm{d}\bar{x}. \tag{A.4}$$

The problem then boils down to differentiating $f(\bar{x})$ with respect to w. We assume without loss of generality that w affects the scaled phase functions F_v for $1 \leq v \leq n_0$. Then $f(\bar{x}) = g(\bar{x})\prod_{v=1}^{n_0} F_v$, where $g(\bar{x})$ captures all terms in Equation (A.2) that do not depend on w. It follows that

$$\frac{\partial f}{\partial w}(\bar{x}) = g(\bar{x})\frac{\partial}{\partial w}\prod_{v=1}^{n_0} F_v = g(\bar{x})\sum_{v=1}^{n_0}\left[\frac{\partial}{\partial w}F_v \prod_{v'\neq v} F_{v'}\right], \tag{A.5}$$

where $\frac{\partial}{\partial w}F_v$ can be obtained via Equations (A.3), (3.9) and (3.13). In practice, we compute Equation (A.4) using unidirectional path tracing.

Appendix B

Appendix for Chapter 4

B.1 Average slope of a bilinear patch

Let $h_{00}, h_{10}, h_{01}, h_{11}$ be the height values at the four corners of a unit patch $[0,1]^2$. Then, the height at any point (u,v) within the patch can be obtained via a bilinear interpolation:

$$h(u,v) = (1-u)(1-v)h_{00} + u(1-v)h_{10} + v(1-u)h_{01} + uvh_{11}. \tag{B.1}$$

The slope at each point can be computed as

$$x_s(u,v) = \frac{h(u+\Delta u, v) - h(u,v)}{\Delta u} = h_{10} - h_{00} + v(h_{00} + h_{11} - h_{10} - h_{01}), \tag{B.2}$$

$$y_s(u,v) = \frac{h(u, v+\Delta v) - h(u,v)}{\Delta v} = h_{01} - h_{00} + u(h_{00} + h_{11} - h_{10} - h_{01}). \tag{B.3}$$

So the average slope of the bilinear patch is

$$x_{\tilde{s}} = \int_0^1 \int_0^1 x_s(u,v)\,\mathrm{d}u\,\mathrm{d}v = \frac{1}{2}(h_{11} + h_{10} - h_{01} - h_{00}), \tag{B.4}$$

$$y_{\tilde{s}} = \int_0^1 \int_0^1 y_s(u,v)\,\mathrm{d}u\,\mathrm{d}v = \frac{1}{2}(h_{11} + h_{01} - h_{10} - h_{00}). \tag{B.5}$$

B.2 Factorization of R_{ir}

Expanding $T_{\text{ir}}(\boldsymbol{x})$ as Equations (4.22), (4.23) and $S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ as Equation (4.24) gives

$$\begin{aligned} T_{\text{ir}}(\boldsymbol{x}) S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = & \frac{\int_{\mathcal{H}^2}\int_{\mathcal{H}^2} R_{\text{ir}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i\,\mathrm{d}\boldsymbol{\omega}_o}{\int_{\mathcal{P}_{\text{all}}}\int_{\mathcal{H}^2}\int_{\mathcal{H}^2} R_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{\omega}_i\,\mathrm{d}\boldsymbol{\omega}_o\,\mathrm{d}\boldsymbol{p}} \\ & \times \int_{\mathcal{P}_{\text{all}}} R_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p} \end{aligned} \tag{B.6}$$

As $R_{\text{ir}}(\boldsymbol{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \approx T_{\text{ir}}(\boldsymbol{x}) \cdot S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$, we can simplify this formula by factoring out R_{ir} and regrouping the terms:

$$\begin{aligned} & \frac{T_{\text{ir}}(\boldsymbol{x}) \int_{\mathcal{H}^2}\int_{\mathcal{H}^2} S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i\,\mathrm{d}\boldsymbol{\omega}_o}{\int_{\mathcal{P}_{\text{all}}} T_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p})) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p}) \left[\int_{\mathcal{H}^2}\int_{\mathcal{H}^2} S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i\,\mathrm{d}\boldsymbol{\omega}_o\right]\mathrm{d}\boldsymbol{p}} \\ & \times S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \int_{\mathcal{P}_{\text{all}}} T_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p})) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p} \\ = & \frac{T_{\text{ir}}(\boldsymbol{x}) \int_{\mathcal{H}^2}\int_{\mathcal{H}^2} S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i\,\mathrm{d}\boldsymbol{\omega}_o}{\left[\int_{\mathcal{P}_{\text{all}}} T_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p})) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p}\right] \left[\int_{\mathcal{H}^2}\int_{\mathcal{H}^2} S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i\,\mathrm{d}\boldsymbol{\omega}_o\right]} \\ & \times S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \int_{\mathcal{P}_{\text{all}}} T_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p})) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p} \\ = & \frac{T_{\text{ir}}(\boldsymbol{x})}{\int_{\mathcal{P}_{\text{all}}} T_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p})) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p}} S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \int_{\mathcal{P}_{\text{all}}} T_{\text{ir}}(\boldsymbol{x}_m(\boldsymbol{p})) k_{\mathcal{P}_{\text{all}}}(\boldsymbol{p})\,\mathrm{d}\boldsymbol{p} \\ = & T_{\text{ir}}(\boldsymbol{x}) S_{\text{ir}}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o). \end{aligned} \tag{B.7}$$

Therefore, the rank-1 factorization is consistent with our definition of T_{ir} and S_{ir}.

Appendix C

Appendix for Chapter 5

C.1 Sharing ZH Lobes Across Bands

To enable efficient ZH factorization, Nowrouzezahrai et al. [75] presented a ZH lobe sharing strategy. For SH basis functions up to degree $l_{\max}$, the central directions of ZH lobes are

$$\begin{aligned}\{\boldsymbol{\omega}_{0,0} = \boldsymbol{\omega}_{1,0} = \boldsymbol{\omega}_{2,0} = \boldsymbol{\omega}_{3,0} = \cdots = \boldsymbol{\omega}_{l_{\max},0},\\ \boldsymbol{\omega}_{1,1} = \boldsymbol{\omega}_{2,1} = \boldsymbol{\omega}_{3,1} = \cdots = \boldsymbol{\omega}_{l_{\max},1},\\ \boldsymbol{\omega}_{1,2} = \boldsymbol{\omega}_{2,2} = \boldsymbol{\omega}_{3,2} = \cdots = \boldsymbol{\omega}_{l_{\max},2},\\ \ldots\ldots, \boldsymbol{\omega}_{l_{\max},2l_{\max}}\}.\end{aligned} \tag{C.1}$$

Specifically, the band-l SH will use a set of $2l+1$ central directions $\{\boldsymbol{\omega}_{l,0}, \ldots, \boldsymbol{\omega}_{l,2l}\}$.

C.2 Relation to Prior Work [1]

Annen et al. [1] developed a semi-analytic solution to SH gradients. They integrate the SH basis functions over the area domain A of the polygonal light source, which is independent of the varying shading point $\boldsymbol{x}$. By changing the solid angle measure to the area measure in Equation (5.4), the SH coefficients can be rewritten as

$$L_{lm}(\boldsymbol{x}) = \int_{[0,1]^2} Y_{lm}(\boldsymbol{s}(\boldsymbol{u})) \frac{\langle \boldsymbol{n}(\boldsymbol{y}(\boldsymbol{u})), -\boldsymbol{s}(\boldsymbol{u})\rangle}{\|\boldsymbol{y}(\boldsymbol{u}) - \boldsymbol{x}\|^2} |\det \mathcal{J}_{\boldsymbol{y}}| \, \mathrm{d}\boldsymbol{u}. \tag{C.2}$$

Note that $\boldsymbol{y} : [0,1]^2 \to A$ is a transformation that warps a unit square to a polygon and $\mathcal{J}_{\boldsymbol{y}}$ denotes the corresponding Jacobian matrix. The normalized direction vector $\boldsymbol{s}(\boldsymbol{u}) = \frac{\boldsymbol{y}(\boldsymbol{u})-\boldsymbol{x}}{\|\boldsymbol{y}(\boldsymbol{u})-\boldsymbol{x}\|}$ is from the shading point $\boldsymbol{x}$ to a point $\boldsymbol{y}(\boldsymbol{u})$ on the area light. In the change-of-measure term $\frac{\langle \boldsymbol{n}(\boldsymbol{y}(\boldsymbol{u})), -\boldsymbol{s}(\boldsymbol{u})\rangle}{\|\boldsymbol{y}(\boldsymbol{u})-\boldsymbol{x}\|^2}$, we indicate $\boldsymbol{n}(\boldsymbol{y}(\boldsymbol{u}))$ as the surface normal.

To differentiate the integral in Equation (C.2), we can directly move the differentiation operator into the integration,

$$\partial_z L_{lm}(\boldsymbol{x}) = \int_{[0,1]^2} \partial_z \left[Y_{lm}(\boldsymbol{s}(\boldsymbol{u})) \frac{\langle \boldsymbol{n}(\boldsymbol{y}(\boldsymbol{u})), -\boldsymbol{s}(\boldsymbol{u})\rangle}{\|\boldsymbol{y}(\boldsymbol{u}) - \boldsymbol{x}\|^2} |\det \mathcal{J}_{\boldsymbol{y}}| \right] \mathrm{d}\boldsymbol{u}. \tag{C.3}$$

This is a special case of the Reynolds transport theorem. The boundary integral vanishes since the integration domain is static. The integrand in Equation (C.3) can be computed either analytically or using automatic differentiation. But the 2D integral needs to be evaluated numerically, and has no known analytic form.

C.3 Detailed Derivations

C.3.1 Deriving $\ell(t)$ in Equation (5.17)

After transforming the edge $\overline{\boldsymbol{p}_i \boldsymbol{p}_{i+1}}$ into the local frame $(\boldsymbol{\omega}_i, \boldsymbol{\lambda}_i, \boldsymbol{n}_i)$, the edge is in the xy-plane so we can omit the z-coordinate and solve for $\ell(t)$ in 2D. The 2D local coordinates of the edge endpoints are $\boldsymbol{p}_i = (\ell_i, 0)$ and $\boldsymbol{p}_{i+1} = (\ell_{i+1}\cos T_i, \ell_{i+1}\sin T_i)$. We want to know the travel distance ℓ of the ray with direction $(\cos t, \sin t)$ starting from $(0,0)$, before hitting the edge. The geometric relation can be described in the following linear system,

$$\begin{cases} \ell \cos t = (1-k)\ell_i + k\ell_{i+1}\cos T_i, \\ \ell \sin t = k\ell_{i+1}\sin T_i. \end{cases} \tag{C.4}$$

After eliminating k, we have

$$\ell(\ell_i \sin t - \ell_{i+1} \sin(t - T_i)) = \ell_i \ell_{i+1} \sin T_i$$
$$\Rightarrow \ell(t) = \left(\frac{\sin t}{\ell_{i+1}} - \frac{\sin(t - T_i)}{\ell_i} \right)^{-1} \sin T_i. \tag{C.5}$$

C.3.2 Deriving Equation (5.22)

Let $h = c_x \cos t + c_y \sin t = A \sin(t + T')$, where $A = \sqrt{c_x^2 + c_y^2}$ and $T' = \arctan(c_x / c_y)$. We use a change of variable $u = t + T'$. Then, the first integral on the RHS of Equation (5.20) can be rewritten as

$$\int_0^{T_i} \sin(t - T_i) P_l(c_x \cos t + c_y \sin t)\, \mathrm{d}t = \frac{1}{A} \int_{T'}^{T_i + T'} A \sin(u - (T_i + T')) P_l(A \sin u)\, \mathrm{d}u \tag{C.6}$$

Using the angle difference identity for sine, the formula becomes

$$\frac{1}{A} \int_{T'}^{T_i + T'} A \sin(u) \cos(T_i + T') P_l(A \sin u)\, \mathrm{d}u - \frac{1}{A} \int_{T'}^{T_i + T'} A \cos(u) \sin(T_i + T') P_l(A \sin u)\, \mathrm{d}u. \tag{C.7}$$

Since $h = A \sin(u)$ and $u = t + T'$, we know that $\frac{\mathrm{d}}{\mathrm{d}u} h = A \cos(u)$ and $\frac{\mathrm{d}}{\mathrm{d}t} u = 1$. Therefore, the formula can be further simplified as

$$\begin{aligned} &\frac{\cos(T_i + T')}{A} \int_{T'}^{T_i + T'} h P_l(h)\, \mathrm{d}u - \frac{\sin(T_i + T')}{A} \int_{T'}^{T_i + T'} (\frac{\mathrm{d}}{\mathrm{d}u} h) P_l(h)\, \mathrm{d}u \\ = &\frac{\cos(T_i + T')}{A} \underbrace{\int_0^{T_i} h P_l(h)\, \mathrm{d}t}_{=: C_l} - \frac{\sin(T_i + T')}{A} \underbrace{\int_0^{T_i} (\frac{\mathrm{d}}{\mathrm{d}t} h) P_l(h)\, \mathrm{d}t}_{=: E_l}. \end{aligned} \tag{C.8}$$

Using the same technique, the second integral on the RHS of Equation (5.20) can be

expressed as

$$
\begin{aligned}
&\int_0^{T_i} \sin(t) P_l(c_x \cos t + c_y \sin t)\,\mathrm{d}t \\
&= \frac{1}{A}\int_{T'}^{T_i+T'} A\sin(u-T') P_l(A\sin u)\,\mathrm{d}u \\
&= \frac{1}{A}\int_{T'}^{T_i+T'} A\sin(u)\cos(T') P_l(A\sin u)\,\mathrm{d}u \;-\; \frac{1}{A}\int_{T'}^{T_i+T'} A\cos(u)\sin(T') P_l(A\sin u)\,\mathrm{d}u \\
&= \frac{\cos(T')}{A}\underbrace{\int_0^{T_i} h P_l(h)\,\mathrm{d}t}_{=:C_l} \;-\; \frac{\sin(T')}{A}\underbrace{\int_0^{T_i} (\frac{\mathrm{d}}{\mathrm{d}t}h) P_l(h)\,\mathrm{d}t}_{=:E_l}.
\end{aligned}
\tag{C.9}
$$

Bibliography

[1] Thomas Annen, Jan Kautz, Frédo Durand, and Hans-Peter Seidel. Spherical Harmonic Gradients for Mid-Range Illumination. *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*, pages 331–336, 2004.

[2] James Arvo. The Irradiance Jacobian for Partially Occluded Polyhedral Scenes. *SIGGRAPH*, pages 343–350, 1994.

[3] James Arvo. Applications of Irradiance Tensors to the Simulation of Non-Lambertian Phenomena. *SIGGRAPH*, pages 335–342, 1995.

[4] Michael Ashikhmin and Peter Shirley. An Anisotropic Phong BRDF Model. *Journal of Graphics Tools*, 5(2):25–32, 2000.

[5] Laurent Belcour, Guofu Xie, Christophe Hery, Mark Meyer, Wojciech Jarosz, and Derek Nowrouzezahrai. Integrating Clipped Spherical Harmonics Expansions. *ACM Transactions on Graphics*, 37(2):19:1–19:12, 2018.

[6] Nir Benty, Kai-Hwa Yao, Petrik Clarberg, Lucy Chen, Simon Kallweit, Tim Foley, Matthew Oakes, Conor Lavelle, and Chris Wyman. The Falcor Rendering Framework, 2020. https://github.com/NVIDIAGameWorks/Falcor.

[7] Eric Bruneton and Fabrice Neyret. A Survey of Nonlinear Prefiltering Methods for Efficient and Accurate Surface Shading. *IEEE TVCG*, 18(2):242–260, 2012.

[8] Brian Cabral, Nelson Max, and Rebecca Springmeyer. Bidirectional Reflection Functions from Surface Bump Maps. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 273–281, 1987.

[9] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Dover Publications Inc., 1960.

[10] Min Chen and James Arvo. A Closed-Form Solution for the Irradiance Due to Linearly-Varying Luminaires. *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*, pages 137–148, 2000.

[11] Min Chen and James Arvo. Simulating Non-Lambertian Phenomena Involving Linearly-Varying Luminaires. *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*, pages 25–38, 2001.

[12] Xavier Chermain, Frédéric Claux, and Stéphane Mérillou. A Microfacet-Based BRDF for the Accurate and Efficient Rendering of High-Definition Specular Normal Maps. *The Visual Computer (Proceedings of CGI)*, 2018.

[13] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):1166–1175, 2005.

[14] Robert L Cook, John Halstead, Maxwell Planck, and David Ryu. Stochastic Simplification of Aggregate Detail. *ACM Transactions on Graphics*, 26(3):79:1–79:8, 2007.

[15] Robert L Cook and Kenneth E Torrance. A Reflectance Model for Computer Graphics. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 7–24, 1982.

[16] Kristin J Dana, Bram Van Ginneken, Shree K Nayar, and Jan J Koenderink. Reflectance and Texture of Real-World Surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.

[17] Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. Linear Efficient Antialiased Displacement and Reflectance Mapping. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 32(6):211:1–211:11, 2013.

[18] Alain Fournier. Normal Distribution Functions and Multiple Surfaces. *Graphics Interface '92 Workshop on Local Illumination*, pages 45–52, 1992.

[19] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light Transport Simulation with Vertex Connection and Merging. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 31(6):192:1–192:10, 2012.

[20] Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. Sample-Based Monte Carlo Denoising Using a Kernel-Splatting Network. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 38(4):125:1–125:12, 2019.

[21] Ioannis Gkioulekas, Anat Levin, and Todd Zickler. An Evaluation of Computational Imaging Techniques for Heterogeneous Inverse Scattering. In *European Conference on Computer Vision*, pages 685–701, 2016.

[22] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse Volume Rendering with Material Dictionaries. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 32(6):162:1–162:13, 2013.

[23] Gene Greger, Peter Shirley, Philip M Hubbard, and Donald P Greenberg. The Irradiance Volume. *IEEE Computer Graphics and Applications*, 18:32–43, 1998.

[24] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. A Path Space Extension for Robust Light Transport Simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 31(6):191:1–191:10, 2012.

[25] Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. Frequency Domain Normal Map Filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 26(3):28:1–28:12, 2007.

[26] Milovš Hašan and Ravi Ramamoorthi. Interactive Albedo Editing in Path-traced Volumetric Materials. *ACM Transactions on Graphics*, 32(2):11:1–11:11, 2013.

[27] Wolfgang Heidrich, Katja Daubert, Jan Kautz, and Hans-Peter Seidel. Illuminating Micro Geometry Based on Precomputed Visibility. *SIGGRAPH*, pages 455–464, 2000.

[28] Eric Heitz. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs. *Journal of Computer Graphics Techniques*, 3(2):32–91, 2014.

[29] Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. The SGGX Microflake Distribution. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 34(4):48:1–48:11, 2015.

[30] Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. Real-Time Polygonal-Light Shading with Linearly Transformed Cosines. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(4):41:1–41:8, 2016.

[31] Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher. Multiple-Scattering Microfacet BSDFs with the Smith Model. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(4):58:1–58:14, 2016.

[32] Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. Product Importance Sampling for Light Transport Path Guiding. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 35(4):67–77, 2016.

[33] Nicolas Holzschuch and François Sillion. Accurate Computation of the Radiosity Gradient with Constant and Linear Emitters. *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*, pages 186–195, 1995.

[34] Nicolas Holzschuch and François Sillion. An Exhaustive Error-Bounding Algorithm for Hierarchical Radiosity. *Computer Graphics Forum*, 17:197–218, 1998.

[35] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley,

and Frédo Durand. DiffTaichi: Differentiable Programming for Physical Simulation. In *International Conference on Learning Representations*, 2019.

[36] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: A Language for High-Performance Computation on Spatially Sparse Data Structures. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6):201:1–201:16, 2019.

[37] Homan Igehy. Tracing Ray Differentials. *SIGGRAPH*, pages 179–186, 1999.

[38] Teturo Inui, Yukito Tanabe, and Yositaka Onodera. *Group Theory and Its Applications in Physics*. Springer-Verlag, 1990.

[39] Akira Ishimaru. *Wave Propagation and Scattering in Random Media*, volume 2. Academic press New York, 1978.

[40] Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. Interactive Bi-Scale Editing of Highly Glossy Materials. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 31(6):144:1–144:7, 2012.

[41] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[42] Wenzel Jakob, Adam Arbree, Jonathan T Moon, Kavita Bala, and Steve Marschner. A Radiative Transfer Framework for Rendering Materials with Anisotropic Structure. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 29(4):53:1–53:13, 2010.

[43] Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. Discrete Stochastic Microfacet Models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):115:1–115:10, 2014.

[44] Adrian Jarabo, Hongzhi Wu, Julie Dorsey, Holly Rushmeier, and Diego Gutierrez. Effects of Approximate Filtering on the Appearance of Bidirectional Texture Functions. *IEEE TVCG*, 20(6):880–892, 2014.

[45] Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. Radiance Caching for Participating Media. *ACM Transactions on Graphics*, 27(1):7:1–7:11, 2008.

[46] Wojciech Jarosz, Volker Schönefeld, Leif Kobbelt, and Henrik Wann Jensen. Theory, Analysis and Applications of 2D Global Illumination. *ACM Transactions on Graphics*, 31(5):125:1–125:21, 2012.

[47] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. Irradiance Gradients in the Presence of Participating Media and Occlusions. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 27(4):1087–1096, 2008.

[48] James T Kajiya. The Rendering Equation. *Computer Graphics (Proceedings of SIGGRAPH)*, 20(4):143–150, 1986.

[49] Anton S Kaplanyan, Stephen Hill, Anjul Patney, and Aaron E Lefohn. Filtering Distributions of Normals for Shading Antialiasing. *Proceedings of High Performance Graphics*, pages 151–162, 2016.

[50] Pramook Khungurn, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner. Matching Real Fabrics with Micro-Appearance Models. *ACM Transactions on Graphics*, 35(1):1:1–1:26, 2015.

[51] Martin Kraus and Kai Bürger. Interpolating and Downsampling RGBA Volume Data. *Proceedings of Vision, Modeling and Visualization*, pages 323–332, 2008.

[52] Jaroslav Křivánek, Kadi Bouatouch, Sumanta Pattanaik, and Jiří Žára. Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping. *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*, pages 127–138, 2006.

[53] Jaroslav Křivánek, Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Improved Radiance Gradient Computation. *SCCG '05: Proceedings of the 21th spring conference on computer graphics*, pages 155–159, 2005.

[54] Jaroslav Křivánek, Pascal Gautron, Greg Ward, Henrik Wann Jensen, Eric Tabellion, and Per H Christensen. Practical Global Illumination with Irradiance Caching. In *SIGGRAPH Courses*, 2008.

[55] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance Caching for Efficient Global Illumination Computation. *IEEE TVCG*, 11(5):550–561, 2005.

[56] Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. Unifying Points, Beams, and Paths in Volumetric Light Transport Simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):103:1–103:13, 2014.

[57] L Gary Leal. *Advanced Transport Phenomena: fluid mechanics and convective transport processes*, volume 7. Cambridge University Press, 2007.

[58] Pascal Lecocq, Arthur Dufay, Gael Sourimant, and Jean-Eudes Marvie. Analytic Approximations for Real-Time Area Light Shading. *IEEE TVCG*, 23(5):1428–1441, 2017.

[59] Joo Ho Lee, Adrian Jarabo, Daniel S Jeon, Diego Gutierrez, and Min H Kim. Practical Multiple Scattering for Rough Surfaces. *ACM Transactions on Graphics (Proceedings of*

SIGGRAPH Asia), 37(6):275:1–275:12, 2018.

[60] Jaakko Lehtinen. A Framework for Precomputed and Captured Light Transport. *ACM Transactions on Graphics*, 26(4):13:1–13:22, 2007.

[61] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Monte Carlo Ray Tracing through Edge Sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[62] Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. Anisotropic Gaussian mutations for Metropolis Light Transport Through Hessian-Hamiltonian Dynamics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 34(6):209:1–209:13, 2015.

[63] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond Pixel Norm-Balls: Parametric Adversaries using an Analytically Differentiable Renderer. In *International Conference on Learning Representations*, 2019.

[64] Xinhao Liu, Mitsuru Tanaka, and Masatoshi Okutomi. Noise Level Estimation Using Weak Textured Patches of a Single Noisy Image. *IEEE ICIP*, pages 665–668, 2012.

[65] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6):228:1–228:14, 2019.

[66] Guillaume Loubet and Fabrice Neyret. Hybrid Mesh-Volume LoDs for All-Scale Pre-Filtering of Complex 3D Assets. *Computer Graphics Forum (Proceedings of Eurographics)*, 36(2):431–442, 2017.

[67] Wan-Chun Ma, Sung-Hsiang Chao, Yu-Ting Tseng, Yung-Yu Chuang, Chun-Fa Chang, Bing-Yu Chen, and Ming Ouhyoung. Level-of-Detail Representation of Bidirectional Texture Functions for Real-Time Rendering. *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 187–194, 2005.

[68] Thomas MacRobert. *Spherical Harmonics: An Elementary Treatise on Harmonic Functions with Applications*. Dover Publications, 1948.

[69] Julio Marco, Adrian Jarabo, Wojciech Jarosz, and Diego Gutierrez. Second-Order Occlusion-Aware Volumetric Radiance Caching. *ACM Transactions on Graphics*, 37(2):20:1–20:14, 2018.

[70] Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. Multi-scale Modeling and Rendering of Granular Materials. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 34(4):49:1–

49:13, 2015.

[71] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-Frequency Shadows using Non-Linear Wavelet Lighting Approximation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 22(3):376–381, 2003.

[72] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. Triple Product Wavelet Integrals for All-Frequency Relighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 23(3):475–485, 2004.

[73] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6):203:1–203:17, 2019.

[74] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. Monte Carlo Methods for Volumetric Light Transport Simulation. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)*, 37(2), 2018.

[75] Derek Nowrouzezahrai, Patricio Simari, and Eugene Fiume. Sparse Zonal Harmonic Factorization for Efficient SH Rotation. *ACM Transactions on Graphics*, 31(3):23:1–23:9, 2012.

[76] Derek Nowrouzezahrai and John Snyder. Fast Global Illumination on Dynamic Height Fields. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 28(4):1131–1139, 2009.

[77] Marc Olano and Dan Baker. LEAN Mapping. *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 181–188, 2010.

[78] Michael Oren and Shree K Nayar. Generalization of Lambert's Reflectance Model. *SIGGRAPH*, pages 239–246, 1994.

[79] Jacopo Pantaleoni, Luca Fascione, Martin Hill, and Timo Aila. PantaRay: Fast Ray-Traced Occlusion Caching of Massive Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 29(4), 2010.

[80] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis Light Transport for Participating Media. *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*, pages 11—-22, 2000.

[81] Serban D Porumbescu, Brian Budge, Louis Feng, and Kenneth I Joy. Shell Maps. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):626–633, 2005.

[82] Ravi Ramamoorthi. Precomputation-Based Rendering. *Foundations and Trends in*

Computer Graphics and Vision, 3(4):281–369, 2009.

[83] Ravi Ramamoorthi and Pat Hanrahan. A Signal-Processing Framework for Inverse Rendering. *SIGGRAPH*, page 117–128, 2001.

[84] Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. A First Order Analysis of Lighting, Shading, and Shadows. *ACM Transactions on Graphics*, 26(1), 2007.

[85] Zhong Ren, Rui Wang, John Snyder, Kun Zhou, Xinguo Liu, Bo Sun, Peter-Pike Sloan, Hujun Bao, Qunsheng Peng, and Baining Guo. Real-time Soft Shadows in Dynamic Scenes using Spherical Harmonic Exponentiation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 25(3):977–986, 2006.

[86] Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. Practical Hessian-Based Error Control for Irradiance Caching. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 31(6):193:1–193:10, 2012.

[87] Peter Shirley and Kenneth Chiu. A Low Distortion Map between Disk and Square. *Journal of Graphics Tools*, 2(3):45–52, 1997.

[88] Ronell Sicat, Jens Kruger, Torsten Moller, and Markus Hadwiger. Sparse PDF Volumes for Consistent Multi-Resolution Volume Rendering. *IEEE TVCG*, 20(12):2417–2426, 2014.

[89] 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 (Proceedings of SIGGRAPH)*, 21(3):527–536, 2002.

[90] Bruce Smith. Geometrical Shadowing of a Random Rough Surface. *IEEE Transactions on Antennas and Propagation*, 15(5):668–671, 1967.

[91] John Snyder. Area Light Sources for Real-Time Graphics. *Technical Report MSR-TR-96-11*, 1996.

[92] John Snyder and Derek Nowrouzezahrai. Fast Soft Self-Shadowing on Dynamic Height Fields. *Computer Graphics Forum*, 27(4):1275–1283, Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering).

[93] Bo Sun and Ravi Ramamoorthi. Affine Double and Triple Product Wavelet Integrals for Rendering. *ACM Transactions on Graphics*, 28(2):14:1–14:17, 2009.

[94] Ping Tan, Stephen Lin, Long Quan, Baining Guo, and Harry Shum. Filtering and Rendering of Resolution-Dependent Reflectance Models. *IEEE TVCG*, 14(2):412–425, 2008.

[95] Ping Tan, Stephen Lin, Long Quan, Baining Guo, and Heung-Yeung Shum. Multiresolution Reflectance Filtering. *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*, pages 111–116, 2005.

[96] Ville Timonen and Jan Westerholm. Scalable Height Field Self-Shadowing. *Computer Graphics Forum (Proceedings of Eurographics)*, 29(2):723–731, 2010.

[97] Michael Toksvig. Mipmapping Normal Maps. *Journal of Graphics Tools*, 10(3):65–71, 2005.

[98] Charles F Van Loan. *Introduction to Scientific Computing: A Matrix-Vector Approach Using MATLAB*. Prentice Hall, 1996.

[99] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.

[100] Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. Denoising with Kernel Prediction and Asymmetric Loss Functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 37(4):124:1–124:15, 2018.

[101] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet Models for Refraction through Rough Surfaces. *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*, pages 195–206, 2007.

[102] Jingwen Wang and Ravi Ramamoorthi. Analytic Spherical Harmonic Coefficients for Polygonal Area Lights. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 37(4):54:1–54:11, 2018.

[103] Gregory J Ward and Paul S Heckbert. Irradiance Gradients. *Third Eurographics Workshop on Rendering*, pages 85–98, 1992.

[104] Gregory J Ward, Francis M Rubinstein, and Robert D Clear. A Ray Tracing Solution for Diffuse Interreflection. *Computer Graphics (Proceedings of SIGGRAPH)*, 22(4):85–92, 1988.

[105] Stephen H Westin, James R Arvo, and Kenneth E Torrance. Predicting Reflectance Functions from Complex Surfaces. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 255–264, 1992.

[106] Hongzhi Wu, Julie Dorsey, and Holly Rushmeier. Characteristic Point Maps. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 28(4):1227–1236, 2009.

[107] Hongzhi Wu, Julie Dorsey, and Holly Rushmeier. Physically-Based Interactive Bi-Scale Material Design. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 30(6):145:1–145:10, 2011.

[108] Lifan Wu, Guangyan Cai, Shuang Zhao, and Ravi Ramamoorthi. Analytic Spherical Harmonic Gradients for Real-Time Rendering with Many Polygonal Area Lights. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 39(4):134:1–134:14, 2020.

[109] Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi. Accurate Appearance Preserving Prefiltering for Rendering Displacement-Mapped Surfaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 38(4):137:1–137:14, 2019.

[110] Feng Xie and Pat Hanrahan. Multiple Scattering from Distributions of Specular V-grooves. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 37(6):276:1–276:14, 2018.

[111] Chao Xu, Rui Wang, Shuang Zhao, and Hujun Bao. Real-Time Linear BRDF MIP-Mapping. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 36(4):27–34, 2017.

[112] Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. Anisotropic Spherical Gaussians. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 32(6):209:1–209:11, 2013.

[113] Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. Position-Normal Distributions for Efficient Rendering of Specular Microstructure. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(4):56:1–56:9, 2016.

[114] Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):116:1–116:9, 2014.

[115] Hong Yuan, Derek Nowrouzezahrai, and Peter-Pike Sloan. Irradiance Rigs. *Journal of Graphics, GPU, and Game Tools*, 16(1), 2012.

[116] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-Space Differentiable Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 39(4):143:1–143:19, 2020.

[117] Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. A Differential Theory of Radiative Transfer. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6):227:1–227:16, 2019.

[118] Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. Building Volumetric

Appearance Models of Fabric Using Micro CT Imaging. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 30(4):44:1–44:10, 2011.

[119] Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. Structure-aware Synthesis for Predictive Woven Fabric Appearance. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 31(4):75:1–75:10, 2012.

[120] Shuang Zhao, Ravi Ramamoorthi, and Kavita Bala. High-order Similarity Relations in Radiative Transfer. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):104:1–104:12, 2014.

[121] Shuang Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. Downsampling Scattering Parameters for Rendering Anisotropic Media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 35(6):166:1–166:11, 2016.

[122] Kun Zhou, Yaohua Hu, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Precomputed Shadow Fields for Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):1196–1201, 2005.

[123] Tobias Zirr and Anton S Kaplanyan. Real-Time Rendering of Procedural Multiscale Materials. *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 139–148, 2016.