Title
Physically-based Modeling and Rendering of Complex Visual Appearance

Author
Yan, Lingqi

Publication Date
2018

Peer reviewed|Thesis/dissertation

Physically-based Modeling and Rendering of Complex Visual Appearance

by

Lingqi Yan

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ravi Ramamoorthi, Co-chair
Professor Yi-Ren Ng, Co-chair
Professor Alexei Efros
Professor Martin Banks

Summer 2018

Physically-based Modeling and Rendering of Complex Visual Appearance

Abstract

Physically-based Modeling and Rendering of Complex Visual Appearance

by

Lingqi Yan

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Ravi Ramamoorthi, Co-chair

Professor Yi-Ren Ng, Co-chair

In this dissertation, we focus on physically-based rendering that synthesizes realistic images from 3D models and scenes. State of the art rendering still struggles with two fundamental challenges — realism and speed. The rendered results look artificial and overly perfect, and the rendering process is slow for both offline and interactive applications. Moreover, better realism and faster speed are inherently contradictory, because the computational complexity increases substantially when trying to render higher fidelity detailed results. We put emphasis on both ends of the realism-speed spectrum in rendering by introducing the concept of detailed rendering and appearance modeling to accurately represent and reproduce the rich visual world from micron level to overall appearance, and combining sparse ray sampling with fast high dimensional filtering to achieve real-time performance.

To make rendering more realistic, our first claim is that, we need details. However, rendering a complex surface with lots of details is far from easy. Traditionally, the surface microstructure is approximated using a smooth normal distribution, but this ignores details such as glinty effects, easily observable in the real world. While modeling the actual surface microstructure is possible, the resulting rendering problem is prohibitively expensive using Monte Carlo point sampling: the energy is concentrated in tiny highlights that take up a minuscule fraction of the pixel. We instead compute the accurate solution that Monte Carlo would eventually converge to, using a completely different deterministic approach (Chapter 3). Our method considers the highly complicated distribution of normals on a surface patch seen through a single pixel. We show different methods to evaluate this efficiently with closed-form solutions, assuming a surface patch is made up of either 2D planar triangles [147] or 4D Gaussian elements [145], respectively. We also show how to extend our method to accurately handle wave optics [148]. Our results show complicated, temporally varying glints from materials such as bumpy plastics, brushed and scratched metals, metallic paint and ocean waves.

In the above, although rendering details imposes many challenges, we assumed we know how the surface reflects light. However, there are a lot of natural materials in the real world

where we are not sure exactly how they interact with the light. To render these materials realistically, we need accurate appearance/reflectance models derived from microstructures to define their optical behavior. We demonstrate this by introducing a reflectance model for animal fur in Chapter 4. Rendering photo-realistic animal fur is a long-standing problem in computer graphics. Considerable effort has been made on modeling the geometric complexity of human hair, but the appearance/reflectance of fur fibers is not well understood. Based on anatomical literature and measurements, we develop a double cylinder model for the reflectance of a single fur fiber, where an outer cylinder represents the biological observation of a cortex covered by multiple cuticle layers, and an inner cylinder represents the scattering interior structure known as the medulla, often absent from human hair fibers. We validate our physical model with measurements on real fur fibers, and introduce the first database in computer graphics of reflectance profiles for nine fur samples. For efficient rendering, we develop a method to precompute 2D medulla scattering profiles and analytically approximate our reflectance model with factored lobes [144]. We then develop a number of optimizations that improve efficiency and generality without compromising accuracy [141]. And we present the first global illumination model, based on dipole diffusion for subsurface scattering, to approximate light bouncing between individual fur fibers by modeling complex light and fur interactions as subsurface scattering, and using a simple neural network to convert from fur fibers' properties to scattering parameters [142].

However, even without these details to improve rendered realism, current rendering still suffers from low performance with state of the art Monte Carlo ray tracing. Physically correct, noise-free images can require hundreds or thousands of ray samples per pixel, and take a long time to compute. Recent approaches have exploited sparse sampling and filtering; the filtering is either fast (axis-aligned), but requires more input samples, or needs fewer input samples but is very slow (sheared). We present a new approach for fast sheared filtering on the GPU in Chapter 5 [143]. Our algorithm factors the 4D sheared filter into four 1D filters. We derive complexity bounds for our method, showing that the per-pixel complexity is reduced from $O(n^2l^2)$ to $O(nl)$, where n is the linear filter width (filter size is $O(n^2)$) and l is the (usually very small) number of samples for each dimension of the light or lens per pixel (spp is l^2). We thus reduce sheared filtering overhead dramatically. We demonstrate rendering of depth of field, soft shadows and diffuse global illumination at interactive speeds.

To My Family

Thank you for your encouragement and support for all these years.
You are the world to me.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank Prof. Ravi Ramamoorthi for being such a great advisor. His insights, guidance, availability and responsiveness are invaluable to the completion of this work, as well as to my future career.

I also thank my dissertation committee members: Prof. Ren Ng, Prof. Alexei Efros and Professor Martin S. Banks. Thank you for all the constructive discussions and all the time spent. Special thanks to Prof. Ren Ng for having me as a Graduate Student Instructor (GSI) for the Computer Graphics course and giving me precious opportunities to give lectures.

Furthermore, I would like to thank my academic collaborators. Dr. Milos Hasan, Prof. Fredo Durand, Prof. Henrik Wann Jensen, Prof. Steve Marschner, Prof. Derek Nowrouzezahrai. I have been fortunate to work with such terrific all-star colleagues.

This work cannot be done without the help from other student collaborators, either. Thanks Soham Mehta, Chi-Wei Tseng, Lifan Wu, Alexandr Kuznetsov and Weilun Sun for the excellent collaborations.

I would also like to thank my parents and my wife Yifan Wei for their continuous support over all these years.

This work was supported in part by NSF grants 1011832, 1115242, 1451830 and 1703957, an NVIDIA Ph.D. fellowship, 3D models from Walt Disney Animation Studios, and a gift from AutoDesk to the UC San Diego Center for Visual Computing.

Chapter 1

Introduction

1.1 Overview

Rendering is a fundamental problem in Computer Graphics. It is the process that synthesizes realistic images from 3D models and scenes. Along with other areas in Computer Graphics, including animation, simulation, display and cameras, Rendering is being pervasively used in modern technologies, providing a variety of visual content to the world.

Recently, photorealistic Rendering in Computer Graphics is undergoing a renaissance. In the movie industry, it has already become a general standard to use physically based path tracing to achieve cinematic realism offline. For interactive applications such as video games, breathtaking graphics have also become one of the most crucial factors to their success. And rendering is applied everywhere in people's everyday lives. Most of the commercials show rendered products instead of real objects, such as cars, jewelleries and electronics. Virtual stores allow people to explore realistic cosmetics, bags, furniture, and clothes under different lighting conditions. With the development of rendering technologies, we are stepping into an age where there's nothing real behind the screen.

With the growing popularity of Rendering, its extensive application also brings challenges. For example, in the movie industry, people can still easily judge whether a character is rendered or real. The cost of production rendering is still prohibitively high. And GPUs still cannot be used for production rendering. In Virtual Reality (VR) and Augmented Reality (AR) for mobile devices, high quality and high performance stereo rendering is still difficult. And in video games, real-time ray tracing is still not applicable, even with the rapid development of graphics hardware. Difficult as these problems are, they also lead to interesting research topics and opportunities.

We categorize the fundamental challenges of state of the art rendering into two types — realism and speed. Currently, rendered images often look artificial and overly perfect, and rendering is slow for both offline and interactive applications. Ultimately, we intend to figure out what makes the world look realistic, and present people a virtual world in real time that is indistinguishable from the one we are living in.

Figure 1.1: Detailed rendering from complex surfaces. Note how the glints (scratches and tiny highlights) dramatically improve realism. [55, 145]

In this dissertation, we aim at mathematically modeling and rendering visual appearance at real world complexity, while exploring theory and practical algorithms to make it real-time. This work involves 7 ACM SIGGRAPH, SIGGRAPH Asia, Transactions on Graphics papers on various topics, including detailed rendering [145, 147, 148], appearance modeling [141, 142, 144] and sampling and reconstruction theory [143].

1.2 Detailed Rendering from Complex Surfaces

"For every complex problem, there is an answer that is clear, simple, and wrong.[1]*"* This statement is suitable for most of the current surface models. Traditional rendering techniques represent materials using smooth BRDFs (Bidirectional Reflectance Distribution Functions, as will be introduced in Chapter 2) describing how light is reflected after interacting with these materials. Since they use smooth BRDFs, these techniques generate perfectly smooth appearances. However, the real world is imperfect. Bumps, flakes and dents can be seen everywhere. These details introduce variance and are key to the realism of the appearance (Fig. 1.1).

The smooth BRDF concept has been standard for nearly 4 decades, prior to our work which introduces a more statistical discrete version of the BRDF, and enables rendering of detailed glints from complex surfaces, such as metallic flakes and scratches. These details are either procedurally generated as random processes, or defined using extremely high resolution normal maps specifying the surface normals or facing directions at different places. So, the surfaces are essentially represented using tiny *microfacets* explicitly, as will be introduced in Chapter 2.

However, existing microfacet models [123] use statistics to represent the distribution of surface normals (NDFs). So the NDF is considered as a smooth probability distribution. The smooth NDF results in smooth appearance, eliminating all the details. We compute the

[1] H. L. Mencken

Figure 1.2: Physically-based appearance modeling of animal fur [144, 141, 142].

actual NDF covered by each pixel, namely $\mathcal{P}$-NDF, and introduce a level of detail that was never dealt with in Computer Graphics previously.

To calculate the $\mathcal{P}$-NDFs, we propose two different methods. In Chapter 3, we will first introduce our solution in 2014 [147], which discretizes the normal map into axis-aligned triangles, and analytically integrates each triangle to compute its contribution to any given direction in a $\mathcal{P}$-NDF. Then, we will introduce our 2016 solution [145] which approximates the normal map using Gaussian elements, resulting in over 100× speedup with the same quality.

Furthermore, we introduce wave optics to detailed rendering [148], to correctly generate diffraction effects such as colors from CDs and dull polished metal, which cannot be correctly produced using traditional geometric optics. We provide a physically-based rendering solution to arbitrary surfaces under wave optics. At a high level, the height field describing the surface's heights at different positions changes how far the light travels, so it acts as a spatially-varying phase shift of light waves. And the corresponding wave optics behavior can be calculated by taking a Fourier transform of the phase shift. However, the phase shift has an even higher resolution than the heightfield. So, taking its Fourier transform directly is very inefficient. To deal with the inefficiency, we use Gabor kernels, sine or cosine weighted Gaussian functions, to fit the phase shift. By taking the Fourier transform of these Gabor kernels, we obtain an efficient and accurate analytical solution.

1.3 Physically-based Appearance Modeling

Physically-based rendering is only one aspect of creating photorealistic images. The real world is versatile and filled with millions of types of materials, while renderers only support very limited types of appearance models. Even for mature commercial renderers, the number of supported types of materials is shockingly low, usually around 20, e.g. diffuse, specular, glass and so on. In most of the rendering tasks, it is required that artists mix, blend and texture map these materials, which not only is ad-hoc and unrealistic, but also consumes time and increases cost.

Another aspect of our research [144, 141] is aimed at appearance modeling, to discover and describe how unknown materials interact with the light. Specifically, we study the appearance of animal fur (Fig. 1.2). Previous work modeled individual animal fur fibers using cylinders, either opaque or glass-like. This model works well for human hair, but for most animals, it fails. The reason is that, there are complex micro structures inside animal fur fibers that significantly affect their optical properties. Thus, we looked into these complex structures, and proposed a coaxial double cylinder model — the outer cylinder decides the color and the inner cylinder accurately captures how light scatters inside. Our follow-up work further simplifes the double cylinder model and comes up with a novel far field integration scheme to completely avoid tracing multiple rays against single fur fibers, making the double cylinder model suitable for rasterizers such as OpenGL and DirectX for the first time, so games and VR/AR can benefit from it as well.

To validate our model, we compare with actual measurements. We use a spherical gantry to measure 10 types of animal fur including human hair. The spherical gantry can help us hold a single fur fiber vertically, and change the light and camera directions arbitrarily using its two arms. We fix the light source from behind, and record the fibers' brightness measured at every outgoing direction. Then we use our model to fit the measured data. We demonstrate numerically that our method fits much better than traditional hair models.

In addition to the appearance model for individual fur fibers, one of the key effects in animal fur rendering is global illumination, involving light bouncing between different fibers. This is very time-consuming to simulate with methods like path tracing. Efficient global illumination techniques are in widespread use, but are limited to human hair only, and cannot handle color bleeding, transparency and hair-object inter-reflection. We present the first global illumination model [142], based on dipole diffusion for subsurface scattering (Chapter 2), to approximate light bouncing between individual fur fibers. We model complex light and fur interactions as subsurface scattering, and use a simple neural network to convert from fur fibers' properties to scattering parameters. Our network is trained on only a single scene with different parameters, but applies to general scenes and produces visually accurate appearance, supporting color bleeding and further inter-reflections.

1.4 Real-time Ray Traced Realism

With our detailed rendering and detailed appearance research, we can expect better realism than what was achieved earlier in Computer Graphics. However, even without this level of realism, state of the art Monte Carlo ray tracing, a specific rendering method known for its physical correctness, still suffers from low performance. Noise-free images rendered using Monte Carlo ray tracing often require hundreds or thousands of samples (light paths) per pixel and take a long time to converge. This is also the reason why current video games and other real-time applications use rasterization instead of ray tracing. However, ray tracing outperforms rasterization in terms of quality in a lot of common effects, such as soft shadows, depth of field, and global illumination.

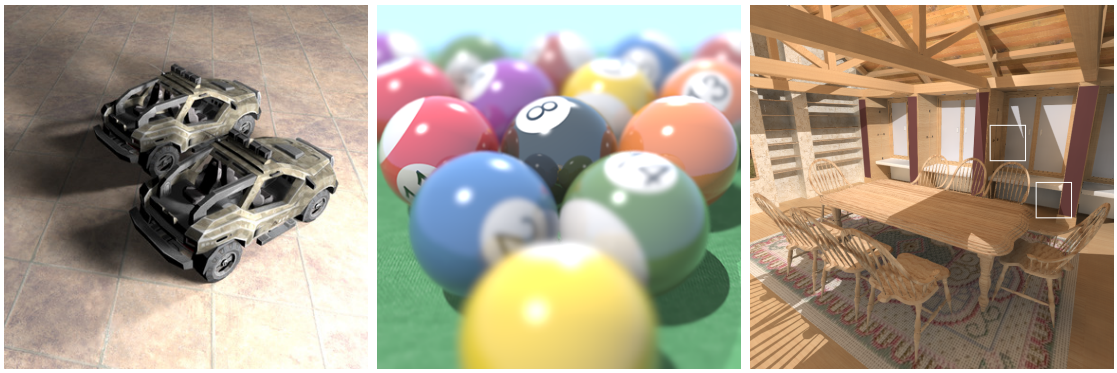


Figure 1.3: Real-time ray tracing and reconstruction of soft shadows, depth of field effects and global illumination [143].

All these effects share the same characteristics, that each pixel is related to many other samples. For example, from each pixel we shoot rays toward different places on the light to calculate partial occlusion and thus soft shadows — this is essentially a 4D light field. Our method [143] is based on new analytic tools of frequency analysis. By analyzing the Fourier spectra of the light field and tightly bounding it using a 4D sheared parallelogram, we know how the light field could be sparsely sampled and filtered in the primal domain. Thus, the sampling rate could be significantly reduced. Then we just need to perform a 4D sheared filter to "clean up" the noisy light field.

The difference between our method and image-based denoising methods is fundamental. The image filtering techniques work on the 2D output image, thus missing information as compared to our 4D light field filter.

However, a 4D sheared filter itself is time-consuming to perform. This is because we need to find all sample points within a 4D space. So, naive implementation of sheared filtering has a time complexity of $O(n^4)$. To make it faster, our idea is that, we need a way to separate the 4D filter into 4 independent 1D filters approximately. At a high level, we first observe that the 4D sheared filter is a product of two 2D sheared filters along orthogonal pixel-sample planes, and develop a two step factored algorithm. We then derive a further factorization of each 2D sheared filter into a pre-convolution and a collection. So, the time complexity of our fast sheared filtering decreases from $O(n^4)$ to $O(n)$.

Our fast sheared filtering work is the first that achieves near real-time performance. With the help of a GPU implemented ray tracer and our fast reconstruction algorithm, the entire process of sampling and filtering can even perform in real time with negligible quality loss (Fig. 1.3).

1.5 Contribution to Computer Graphics

Our research has brought substantial impact to Computer Graphics, in both academia and the industry.

From a conceptual or theoretical perspective, our work on detailed rendering negates the claim that Computer Graphics problems will be automatically solved by simply waiting for improved computing hardware. In [147], we proved that tracing rays to/from microstructure is essentially equivalent to point sampling Dirac delta functions blindly, which takes infinite time in theory. Moreover, the growth of hardware's computational power is far from satisfactory. For example, currently it usually takes over $10,000$ CPU hours to generate one frame in a 4K film. Without technical and algorithmic developments, it will take decades before GPUs are able to generate a 2017-film-level image in real-time (30 frames per second).

In practice, the images that we render are among some of the new visual effects that cannot be produced by other means in Computer Graphics. The excitement of this visual imagery has led to the papers being chosen for the start of the trailer videos at multiple SIGGRAPH conferences, and almost every paper of ours was given considerable press coverage. Moreover, our detailed rendering work [55, 145] has had an entire session at SIGGRAPH devoted to follow-up work since its emergence, and has also been adopted by world leading commercial renderers for production, such as Manuka by Weta Digital, V-Ray by Chaos Group and Autodesk Fusion 360. Our fur appearance model has already been appreciated by the movie industry in less than one year, and has been pervasively used to render animal characters in the recent movie *War for the Planet of the Apes* and is also introduced in the third version of the textbook *Physically Based Rendering: From Theory to Implementation*. Our real-time ray tracing is the first that achieves near real-time performance, and it inspired a lot of follow-up work, including NVIDIAs latest real-time ray tracing technique and hardware that was announced in GDC 2018.

In what follows, we will first introduce the basic background theory in Chapter 2. Then we present our algorithms in detail from Chapters 3 to 5 for detailed rendering, appearance modeling and real-time ray traced realism, respectively. Finally, we conclude our research on both realism and performance, and discuss a broader range of future work and open problems in rendering and generally in Computer Graphics, in Chapter **??**.

Chapter 2

Background

In this chapter, we briefly recap some background knowledge freqently used throughout this dissertation. We first describe fundamental notations in rendering, including statistical materials, classic light transport and wave optics theory. Then we talk about human hair reflectance models for individual hair fibers, together with an approximate model for global illumination between hair fibers. Finally, we introduce the general Fourier analysis and sheared filtering related to our reconstruction scheme.

2.1 Radiometry

In rendering, we use radiometry to study and quantify light energy. We first list common radiometry terminologies in Table 2.1, then describe related notations and methods in the following sections.

Measurement	**Symbol**	**Definition**	**Unit**
Flux / Power	Φ	energy emitted per united time	W
Radiance	L	flux per unit surface area per unit solid angle	$W \cdot m^{-2} \cdot Sr^{-1}$
Irradiance	E	flux received by unit surface area	$W \cdot m^{-2}$
BRDF	f_r	ratio between differential outgoing radiance and differential incident irradiance	Sr^{-1}

Table 2.1: List of common radiometry terminologies.

2.2 Statistical Appearance Models / Materials

Bidirectional Reflectance Distribution Function (BRDF)

The BRDF is an important concept, and it determines how light interacts with surface materials. At a specific surface location, the BRDF is defined as a 4D function of the

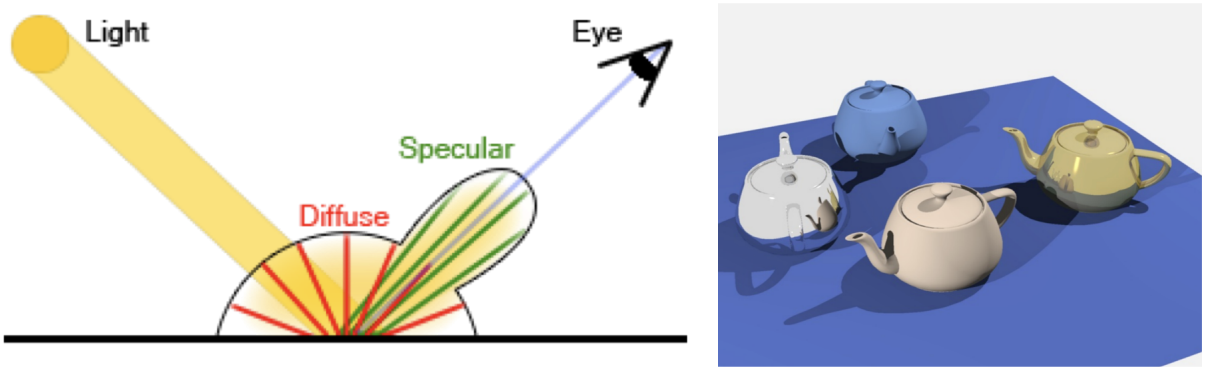


Figure 2.1: (Left) Illustration of the Phong BRDF. (Right) Unrealistic, artificial results generated using the Phong BRDF.

incident and outgoing radiance:

$$f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{\mathrm{d}L_o(\boldsymbol{\omega_o})}{\mathrm{d}E(\boldsymbol{\omega_i})} = \frac{\mathrm{d}L_o(\boldsymbol{\omega_o})}{L_i \cos\theta_i \mathrm{d}\boldsymbol{\omega_i}} \tag{2.1}$$

where f_r is the BRDF, $\boldsymbol{n}$ is the (macro) surface normal, $\boldsymbol{\omega_i}$ and $\boldsymbol{\omega_o}$ are the incident and outgoing directions, and L_i and L_o are the corresponding incident and outgoing radiance, respectively, and $\cos\theta_i = \boldsymbol{n} \cdot \boldsymbol{\omega_i}$.

Different BRDFs determine different types of materials. For example, the Lambertian BRDF $f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = k_d$ spreads the incident energy equally towards different outgoing directions, resulting in diffuse appearance. Here k_d is the albedo or reflectance specifying the absorpion of light thus introducing color. Another example is the Phong BRDF (Figure 2.1) $f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = k_d + k_s(\boldsymbol{r} \cdot \boldsymbol{\omega_o})^p$. It adds a specular component to introduce glossy or shiny appearance. Here $\boldsymbol{r}$ is the reflected incident direction about the normal, and p is an exponential term specifying the extent of shininess.

Note that, the Phong BRDF is an empirical BRDF. This model is based only on observation and is not very accurate and not physically-based. So, it produces unrealistic, artificial results (Figure 2.1).

Microfacet BRDF

State of the art rendering uses an advanced BRDF model, the microfacet BRDF [123].

The microfacet theory assumes that all surfaces are formed by tiny microfacets that are perfectly specular that reflect rays like perfectly smooth mirrors (Figure 2.2 (a)). Each micro facet has a normal, specifying its facing-direction. If the microfacets normals are concentrated around a certain direction, the overall appearance will be glossy. If their normals spread to every direction, the overall surface looks diffuse.

That is to say, the distribution of microfacet normals is crucial to the appearance. We name this distribution as normal distribution function, or NDF. With the NDF, we were able

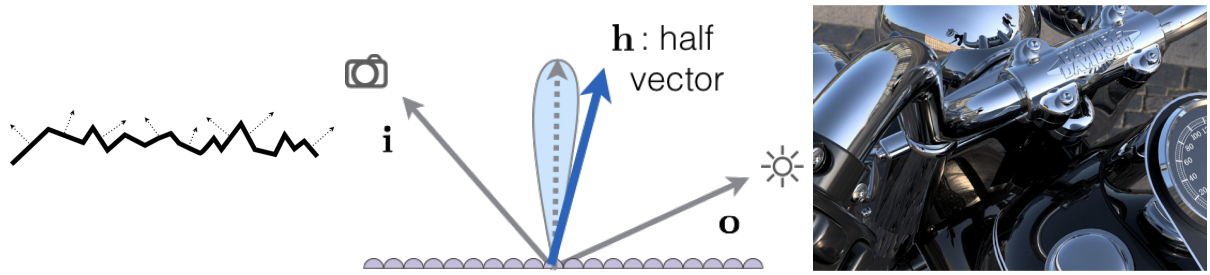


Figure 2.2: (Left) Microfacet surface. (Middle) microfacet NDF. (Right) Pleasing results generated using the microfacet BRDF.

to answer how many microfacets reflect light from the incident direction i to the outgoing direction o, or we can say, how many microfacets' normals point exactly halfway between the camera and light directions, or along the half vector direction $\boldsymbol{h}$ (Figure 2.2 (b)).

The microfacet BRDF uses the NDF as one of its components:

$$f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{F(\boldsymbol{\omega_i}, \boldsymbol{\omega_h}) G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h}) D(\boldsymbol{h})}{4(\boldsymbol{n} \cdot \boldsymbol{\omega_i})(\boldsymbol{n} \cdot \boldsymbol{\omega_o})}, \tag{2.2}$$

which is the D term in this equation. The F term is the Fresnel term that defines the reflectance. And the G term is the shadowing-masking term, to take occluded microfacets into account. These are important but orthogonal to our work. We focus on the NDF or the D term throughout this dissertation.

As shown in Figure 2.2 (c), the microfacet BRDF produces pleasing results. So, they are successfully used in practice.

Subsurface scattering

Instead of being directly reflected at the shading point, light may go into the subsurface, scatter, and exit at a different point, as shown in Fig. 2.3 (a).

The subsurface light transport is often represented using Bidirectional Surface Scattering Reflectance Distribution Functions (BSSRDFs), leading to a generalized reflection equation

$$L_o(x_o, \boldsymbol{\omega_o}) = \int_{\mathcal{A}} \int_{\omega} L_i(x_i, \boldsymbol{\omega_i}) S_{ss}(x_i, \boldsymbol{\omega_i}; x_o, \boldsymbol{\omega_o}) (\boldsymbol{n} \cdot \boldsymbol{\omega_i}) \, \mathrm{d}\boldsymbol{\omega_i} \mathrm{d}\mathcal{A}(x_i), \tag{2.3}$$

where S_{ss} is the BSSRDF, extending a BRDF with different incident and outgoing positions. $\mathcal{A}$ is the area associated with the incident position x_i.

To represent BSSRDFs, material properties that are responsible for the scattering behavior must be defined: σ_a is the absorption coefficient, σ_s is the scattering coefficient, $\sigma_t = \sigma_a + \sigma_s$ is the extinction coefficient, and $\alpha = \sigma_s / \sigma_t$ is the albedo. For anisotropic scattering, an anisotropy factor $g \in [-1, 1]$ is defined, resulting in reduced scattering coefficient $\sigma_s' = (1 - g)\sigma_s$, reduced extinction coefficient $\sigma_t' = \sigma_a + \sigma_s'$ and reduced albedo $\alpha' = \sigma_s' / \sigma_t'$.

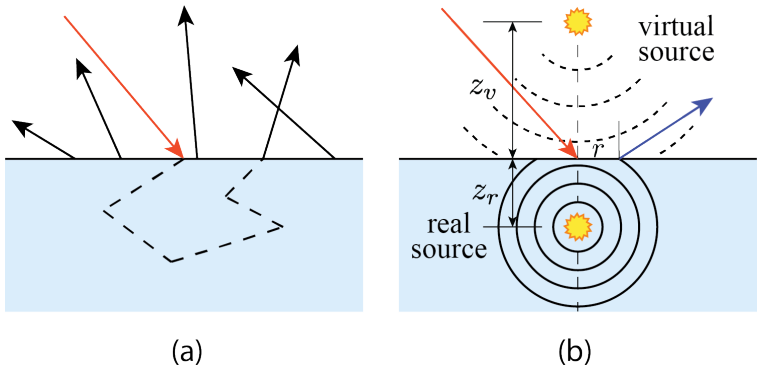


Figure 2.3: (a) Illustration of BSSRDF, where the light exits at different points than the incident point. (b) Dipole method approximating BSSRDF by summing up contribution from a real source and a virtual source to an exiting point at distance r.

While the BSSRDF is often difficult to calculate, Jensen et al. [58] proposed a dipole method to solve the multiple scattering part of it. As illustrated in Fig. 2.3 (b), the dipole method assumes local flatness, putting a real point source beneath the incident x_i and a virtual point source above it. Then the diffuse reflectance at the outgoing position x_o of distance r is the contribution of these dipole sources:

$$R_d(r) = \frac{\alpha' z_r(1+\sigma_{tr}d_r)e^{-\sigma_{tr}d_r}}{4\pi d_r^3} - \frac{\alpha' z_v(1+\sigma_{tr}d_v)e^{-\sigma_{tr}d_v}}{4\pi d_v^3}, \tag{2.4}$$

where $z_r = 1/\sigma_t'$ and $z_v = -z_r(1+4A/3)$ are the positive and negative z-coordinates of the real and virtual point sources, respectively. Here $A = (1+F_{dr})/(1-F_{dr})$, and F_{dr} is a refractive index related variable. $d_r = \sqrt{r^2+z_r^2}$ and $d_v = \sqrt{r^2+z_v^2}$ are distances from x_o to the sources. $\sigma_{tr} = \sqrt{3\sigma_a\sigma_t'}$ is the effective extinction coefficient.

The BSSRDF due to multiple scattering, a.k.a. the diffusion term is then

$$S_d(x_i, \boldsymbol{\omega_i}; x_o, \boldsymbol{\omega_o}) = \frac{1}{\pi} F_t(\eta, \boldsymbol{\omega_i}) R_d(\|x_i - x_o\|) F_t(\eta, \boldsymbol{\omega_o}), \tag{2.5}$$

where F_t terms are for Fresnel transmission.

Apart from the multiple scattered diffusion, to complete the BSSRDF, a single scattering term $S^{(1)}(x_i, \boldsymbol{\omega_i}; x_o, \boldsymbol{\omega_o})$ is added to account for cases where only one scattering event happens. Note that the single-multiple scattering separation is approximate.

To accelerate rendering using the dipole model, Jensen et al. [57] later proposed a two-pass algorithm. In the first pass, they uniformly distribute sample points across the surface of the translucent object. Each sample point's irradiance is evaluated through a path tracing process. The second pass is a traditional path tracing process, where nearby sample points are queried for their contribution to each shading point. To accelerate this process for locating points, a hierarchical octree structure is built on top of these sample points, where each node represents a single sample point, with its irradiance value and position the average

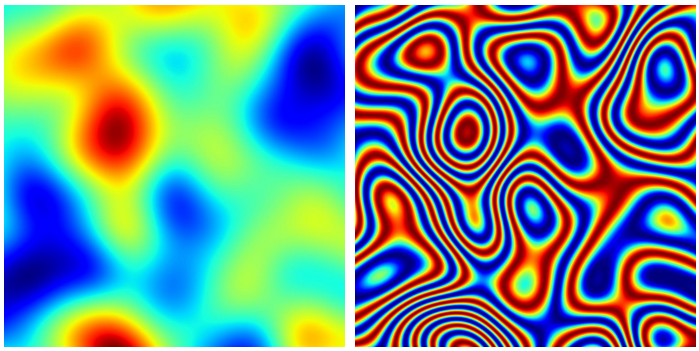


Figure 2.4: Left: A discretized surface heightfield at 1 micron resolution, showing an area of about 64 × 64 microns. For visualization purposes, we complete the heightfield into a continuous function $H(\boldsymbol{s})$ by bicubic interpolation. Right: The real component of the reflection function $R(\boldsymbol{s})$ of this surface patch, specifying the spatially varying phase shift. The imaginary component looks similar.

Diffraction BRDF Model	Equation Components		
	ξ_1	ξ_2	ξ_3
1. OHS	$\frac{\lvert\boldsymbol{\omega}_o \cdot \boldsymbol{n}\rvert\, F}{\lambda^2 \lvert\boldsymbol{\omega}_i \cdot \boldsymbol{n}\rvert}$	1	2
2. GHS	$\frac{\lvert\boldsymbol{\omega}_o \cdot \boldsymbol{n}\rvert\, F}{\lambda^2 \lvert\boldsymbol{\omega}_i \cdot \boldsymbol{n}\rvert}$	1	$\boldsymbol{\psi} \cdot \boldsymbol{n}$
3. R-OHS	$\frac{\lvert\boldsymbol{\psi} \cdot \boldsymbol{n}\rvert^2 F}{4\lambda^2 \lvert\boldsymbol{\omega}_i \cdot \boldsymbol{n}\rvert \lvert\boldsymbol{\omega}_o \cdot \boldsymbol{n}\rvert}$	1	2
4. R-GHS	$\frac{\lvert\boldsymbol{\psi} \cdot \boldsymbol{n}\rvert^2 F}{4\lambda^2 \lvert\boldsymbol{\omega}_i \cdot \boldsymbol{n}\rvert \lvert\boldsymbol{\omega}_o \cdot \boldsymbol{n}\rvert}$	1	$\boldsymbol{\psi} \cdot \boldsymbol{n}$
5. Kirchhoff	$\frac{\lvert\boldsymbol{\psi} \cdot \boldsymbol{n}\rvert^2 F}{4\lambda^2 \lvert\boldsymbol{\omega}_i \cdot \boldsymbol{n}\rvert \lvert\boldsymbol{\omega}_o \cdot \boldsymbol{n}\rvert}$	$1 - \frac{\overline{\boldsymbol{\psi}} \cdot \boldsymbol{H}'(\boldsymbol{s})}{\boldsymbol{\psi} \cdot \boldsymbol{n}}$	$\boldsymbol{\psi} \cdot \boldsymbol{n}$

Figure 2.5: BRDF integrals for five scalar diffraction models (see equations (2.6) and (2.7)). The first two are based on the Original-Harvey-Shack (OHS) and the Generalized-Harvey-Shack (GHS) models. The next two are reciprocal versions of these models we created by substituting Kirchhoff propagation instead of Fourier: Reciprocal OHS (R-OHS) and Reciprocal GHS (R-GHS). The fifth is a fully Kirchhoff-based BRDF model.

of its child nodes. The octree structure is traversed top-down to perform quick rejection of samples that are far enough from the shading point.

2.3 Wave BRDF theory

In wave optics, light is described by fields that satisfy appropriate boundary conditions and governing differential equations (e.g., wave or Helmholtz equations). We will consider each wavelength (denoted λ) separately and use complex-valued fields to encode both magnitude and phase. The local light energy is related to the squared magnitude of the field at that point. Scalar diffraction models, such as Harvey-Shack [64] or Kirchhoff [88], can be used to

estimate the reflected field from a rough surface. Unlike in geometric optics, the contributions from different parts of the surface can sum non-linearly due to interference effects, to create the characteristic diffraction effects of wave optics.

Let us assume we have a surface heightfield $H(\boldsymbol{s})$ such that for a given 2D point $\boldsymbol{s} = [s_x, s_y]$, the corresponding 3D point on the rough surface is $[s_x, s_y, H(\boldsymbol{s})]$. In our approach, the heightfield is typically discretized at the resolution of 1 μm per texel. Figure 2.4 (left) illustrates a small example heightfield. Our goal is to estimate the surface's Bidirectional Reflectance Distribution Function (BRDF) $f_{\mathrm{r}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}})$, which is defined as the ratio between the reflected radiance in direction $\boldsymbol{\omega}_{\mathrm{o}}$ and the incident irradiance from direction $\boldsymbol{\omega}_{\mathrm{i}}$. Light reflecting from different parts of the surface will travel different distances depending on the local surface height. This causes phase shifts in reflected waves which then interfere with each other to determine the BRDF.

These phase shifts can be approximated using a planar surface that reflects light with a spatially-varying phase shift, specified by its reflection function:

$$R(\boldsymbol{s}) = \xi_2 \, e^{-i\frac{2\pi}{\lambda}\xi_3 H(\boldsymbol{s})}. \tag{2.6}$$

Figure 2.4 (right) shows a visualization of the real component of this function. The values of ξ_2 and ξ_3 depend on which diffraction model is chosen (see Figure 2.5 for examples). We represent the directions $\boldsymbol{\omega}_{\mathrm{i}}$ and $\boldsymbol{\omega}_{\mathrm{o}}$ as 3D unit vectors. Let $\boldsymbol{\psi} = \boldsymbol{\omega}_{\mathrm{i}} + \boldsymbol{\omega}_{\mathrm{o}}$ and $\overline{\boldsymbol{\psi}}$ be its 2D projection (by discarding its z-component). The BRDF of this planar proxy can be computed using a surface integral of the form:

$$f_{\mathrm{r}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\xi_1}{A_{\bar{\mathcal{S}}}} \left| \int_{\bar{\mathcal{S}}} R(\boldsymbol{s}) \, e^{-i\frac{2\pi}{\lambda}(\overline{\boldsymbol{\psi}} \cdot \boldsymbol{s})} \, d\boldsymbol{s} \right|^2 \tag{2.7}$$

where $\bar{\mathcal{S}}$ is the domain of the heightfield (i.e. the projection of the rough surface onto the XY plane), $A_{\bar{\mathcal{S}}}$ is its area, and ξ_1 depends on the chosen diffraction model (see Figure 2.5) .

The parameters for five different diffraction models are listed in Figure 2.5. These models are closely related and often produce similar results, especially for low-slope surfaces and paraxial directions. The first four are derived from the Harvey-Shack family of diffraction models [42]. The first uses the phase shift approximation from Original-Harvey-Shack (OHS) and the second uses the more accurate phase shifts from Generalized-Harvey-Shack (GHS). These produce non-reciprocal BRDFs (i.e. $f_{\mathrm{r}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) \neq f_{\mathrm{r}}(\boldsymbol{\omega}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{i}})$). Reciprocal BRDF estimates are often preferred because real world BRDFs are reciprocal, and it also simplifies some light transport algorithms. So we created reciprocal versions (R-OHS and R-GHS) by keeping the same planar proxy and phase shift approximations, but using the Kirchhoff propagation integral instead of their usual Fourier-based propagation. The fifth model is equivalent to the Kirchhoff-derived BRDF in [25] which is also reciprocal.

In Chapter 3, we present an approach that can be used to compute any of these diffraction models.

Coherence area

The spatial size over which the incident light's phase remains correlated (i.e. coherent) is known as its coherence area. Equation 2.7 was derived using incident light with an infinite coherence area, but realistic sources have finite ones (typically inversely related to their solid angle [74]). For example, sunlight [76] has a measured coherence area diameter of roughly one hundred wavelengths, or $\sim$50 microns.

Coherent contributions must be summed using their complex field values, while incoherent ones are best accumulated by summing their energy or averaging their BRDFs. This is commonly simulated (e.g., [71, 25, 138]) by spatially limiting the surface integrals using a coherence kernel $w(\boldsymbol{s})$, and then averaging multiple such BRDF evaluations over the region of interest (e.g., the pixel footprint). The principal effect of limiting the coherence area is a small angular blurring of the BRDF. The BRDF estimate for one coherence area becomes:

$$f_{\mathrm{r}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\xi_1}{A_{\mathrm{c}}} \left| \int_{\bar{\mathcal{S}}_{\mathrm{c}}} R^{\star}(\boldsymbol{s})\, e^{-i\frac{2\pi}{\lambda}(\overline{\boldsymbol{\psi}}\cdot\boldsymbol{s})}\, d\boldsymbol{s} \right|^2 \tag{2.8}$$

$$R^{\star}(\boldsymbol{s}) = w(\boldsymbol{s}-\boldsymbol{x}_{\mathrm{c}})R(\boldsymbol{s}) \tag{2.9}$$

where $\bar{\mathcal{S}}_{\mathrm{c}}$ is the portion of $\bar{\mathcal{S}}$ within the support of the coherence kernel centered at $\boldsymbol{x}_{\mathrm{c}}$, the corresponding normalization factor is $A_{\mathrm{c}} = \int |w(\boldsymbol{s})|^2\, d\boldsymbol{s}$, and $R^{\star}$ is the product of $R(\boldsymbol{s})$ and the coherence kernel. This has the advantages of limiting our integrals to small surface regions and effectively prefiltering the BRDF to remove high frequency angular features that we expect are too small to be resolved. Generally we do not need to exactly match the real coherence area. Overestimating it causes high angular frequencies that can be resolved by using more light samples, while underestimating it causes some angular over-blurring of the BRDF.

During rendering, rather than trying to estimate each source's coherence area, we use a fixed size, which should be at least as large as for any expected light source. For w we use a Gaussian with standard deviation of 10 microns (similar to [138]).

Fourier Interpretation

Let us denote the Fourier transform of a 2D function $f(\boldsymbol{s})$ as:

$$\mathcal{F}\left[f\right](\boldsymbol{v}) \equiv \widetilde{f}(\boldsymbol{v}) \equiv \int_{\mathbb{R}^2} f(\boldsymbol{s})\, e^{-i2\pi(\boldsymbol{s}\cdot\boldsymbol{v})}\, d\boldsymbol{s} \tag{2.10}$$

where $\boldsymbol{v}$ is a 2D frequency vector. Equation 2.8 can be rewritten as:

$$f_{\mathrm{r}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\xi_1}{A_{\mathrm{c}}} \left| \widetilde{R^{\star}}\left(\frac{\overline{\boldsymbol{\psi}}}{\lambda}\right) \right|^2 \tag{2.11}$$

Thus the BRDF can be computed using the Fourier transform of $R^{\star}(\boldsymbol{s})$ evaluated at $\overline{\boldsymbol{\psi}}/\lambda$. One approach could be to compute and store the full Fourier transform, either analytically

or numerically via the Fast Fourier Transform (FFT) algorithm. However we use tabulated heightfields which have no simple analytic Fourier transform, and precomputing FFTs for each surface position would require far too much storage. Computing full FFTs at render time would also be very inefficient as we typically only need one, or at most a few, values for each BRDF evaluation. Also $R^{\star}(\boldsymbol{s})$ typically contains very high frequencies, much higher than those in the original heightfield, so using an FFT would require an extremely fine discretization step of 0.1 microns or less.

2.4 Light Transport

Once we have defined the appearance/materials, the rendering process will find light paths that connect the camera and the light source(s) in the two ends, and bounce between different geometries with various BRDFs. This process is known as *light transport*. Next, we introduce some basic ideas in this topic.

The reflection equation

The reflection equation [60] is a key concept in rendering. It defines the radiance reflected from a surface as a product integral of the incident radiance and the BRDF:

$$L_o(\boldsymbol{\omega_o}) = \int_{\omega} L_i(\boldsymbol{\omega_i}) f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})(\boldsymbol{n} \cdot \boldsymbol{\omega_i}) \, \mathrm{d}\boldsymbol{\omega_i}, \tag{2.12}$$

With the reflection equation, accurate physically-based light transport from the light source(s) to the camera becomes possible. Various rendering techniques attempt to solve the reflection equation, and they can be generalized into two categories: radiosity and ray tracing. In this dissertation, we mainly focus on ray tracing, especially path tracing, which is the most popular ray tracing method.

Note that, the reflection equation is only valid under geometric optics approximation. We will introduce the extended reflection equation for wave optics in Chapter 3.

Path tracing

Path tracing utilizes Monte Carlo sampling to accurately solve the reflection equation [128]. It is a recursive algorithm. For each pixel, it shoots random paths through the pixel, and calculates the intersection with the scene. At each intersection, it randomly samples on the light for *local illumination* and randomly chooses the next direction that the path bounces to for *global illumination*. And this procedure is repeated for all further bounces.

Path tracing is a physically and mathematically correct method that generates high fidelity results under geometric optics. It also natually handles lens effects and volumetric effects such as from cloud, fog and smoke.

However, in order to get high quality images, path tracing requires a very large number of samples for the rendered results to converge to noise-free images. This is a fundamental problem that limits path tracing to work only for offline applications, where long running time is tolerable. Our research is aimed at combining the high quality of path tracing with high performance. In Chapter 5, we show that with properly defined filtering as a post process, it is possible for us to expect ray traced realism in near real-time, and real-time performance at 30 frames per second is getting there.

2.5 Human Hair Reflectance Models

We have briefly introduced the concept of BRDF and light transport. It is worth pointing it out that these models are for surfaces. In Chapter 4, we will introduce another kind of appearance/reflectance model that is defined for thin cylinders with negligible radius (or essentially line segments). This results in slightly different concepts. More specifically, human hair reflectance models treat hair fibers as cylinders, and use the Bidirectional Curve Scattering Distribution Function (BCSDF) to represent reflectance properties of a fiber, resulting in a modified reflection equation:

$$L_r(\omega_r) = \int L_i(\omega_i) S(\omega_i, \omega_r) \cos\theta_i \, \mathrm{d}\omega_i, \tag{2.13}$$

where L_i and L_r are the incoming radiance from direction ω_i, and outgoing radiance in direction ω_r respectively, and S is the BCSDF.

As shown in Fig. 2.6, we follow the longitudinal-azimuthal (θ, ϕ) parameterization in [75],

$$L_r(\theta_r, \phi_r) = \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} L_i(\theta_i, \phi_i) S(\theta_i, \theta_r, \phi_i, \phi_r) \cos^2\theta_i \, \mathrm{d}\theta_i \mathrm{d}\phi_i, \tag{2.14}$$

where the single cosine term becomes squared because the solid angle $\mathrm{d}\omega_i = \cos\theta_i \, \mathrm{d}\theta_i \mathrm{d}\phi_i$ in this parameterization.

Note that, since the hair reflectance models often involve complex light scattering within individual hair fibers, those reflectance models are usually called scattering models as well.

Kajiya-Kay model

The Kajiya-Kay model considers hair fibers as opaque solid cylinders. The reflectance is separated into a diffuse component and a specular component. Following [152], the BCSDF is

$$S(\theta_i, \theta_r, \phi_i, \phi_r) = k_d + k_s \frac{\cos^n(\theta_r + \theta_i)}{\cos\theta_i} \tag{2.15}$$

where k_d and k_s are diffuse and specular coefficients respectively. Note that the Kajiya-Kay model is azimuthally independent.

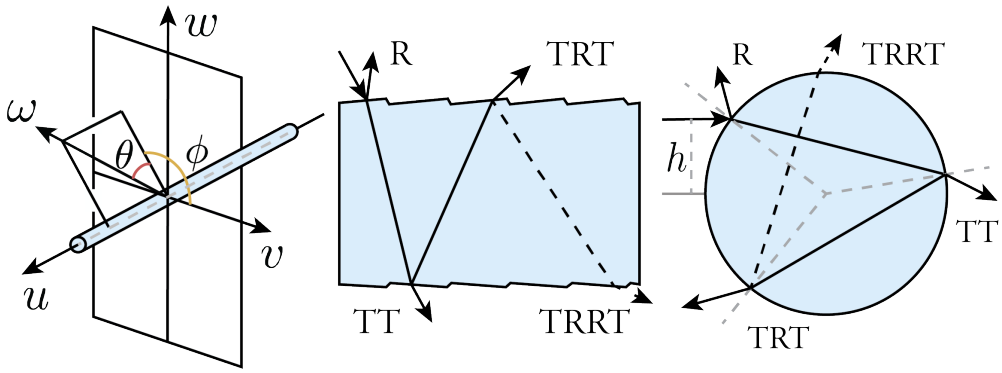


Figure 2.6: (Left) Longitudinal-azimuthal parameterization for hair/fur fibers. θ is angle to plane orthogonal to cylinder axis; ϕ is angle within plane. (Middle & Right) Illustration of Marschner model with factored representation longitudinally and azimuthally.

Marschner model

The BCSDF proposed by Marschner et al. [75] regards hair fibers as glass-like dielectric cylinders. As shown in Fig. 2.6, it takes different specular paths $p \in R, TT, TRT$ into consideration, where R stands for reflection and T for transmission. The contribution of p is factored into a product of M and N profiles, representing longitudinal and azimuthal events:

$$\begin{aligned} S(\theta_i, \theta_r, \phi_i, \phi_r) &= \sum_p S_p(\theta_i, \theta_r, \phi_i, \phi_r) / \cos^2 \theta_d \\ &= \sum_p M_p(\theta_h) \cdot N_p(\phi; \eta') / \cos^2 \theta_d. \end{aligned} \tag{2.16}$$

where $\theta_h = (\theta_r + \theta_i)/2$ and $\theta_d = (\theta_r - \theta_i)/2$ are the longitudinal half angle and difference angle respectively, $\phi = \phi_r - \phi_i$ is the relative outgoing azimuth and $\eta' = \sqrt{\eta^2 - \sin^2 \theta_d} / \cos \theta_d$ is the cortex's virtual refractive index, accounting for inclined longitudinal incident directions. The types of specular paths were later extended by d'Eon et al. [23] to handle multiple internal reflection events such as $TRRT$, etc. Also, since the original Marschner model [75] suffers energy conservation issues addressed by d'Eon et al. [23], we regard [23] as a correct implementation of the Marschner model in all our renderings and comparisons, while we keep the name Marschner model throughout the dissertation.

Near/far field scattering models

In Computer Graphics, especially in the context of hair/fur rendering, it is often convenient to divide the scattering models into near and far field models. The key difference is whether an object can be considered small enough so that positional differences on this object can be ignored. For hair and fur, near-field scattering specifies an actual offset h azimuthally as the incoming position (Fig. 2.6). For far field approximation, parallel light is assumed, covering

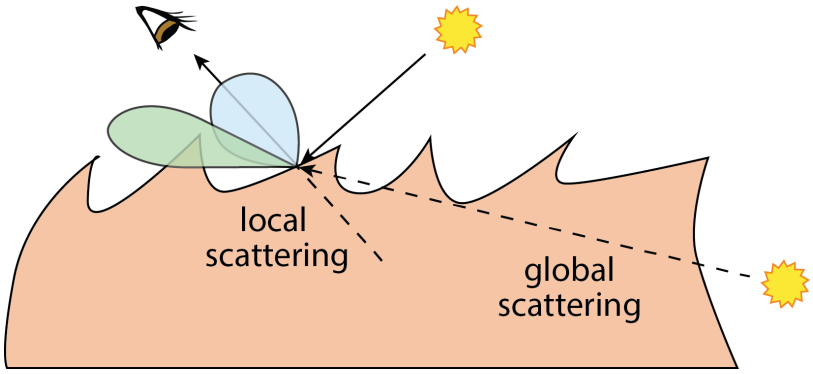


Figure 2.7: (a) Illustration of dual scattering for approximate global illumination. (b) Attenuation and spread computation for global scattering. (c) Attenuation and spread computation for local scattering.

a fiber's width. Thus, far field approximation yields the azimuthal scattering function N_p by integrating over all possible offsets $h \in [-1, 1]$,

$$N_p(\phi) = \frac{1}{2}\int_{-1}^{1} N_p(h, \phi; \eta')\, dh \tag{2.17}$$

The integral for far field approximation could be solved either analytically [75] but only for simple types of paths, or numerically [23] but with more computation.

Dual scattering approximation

In addition to appearance/reflectance models for individual hair/fur fibers, light transport for hair and fur requires simulating global illumination, i.e. multiple light bounces among different hair/fur fibers. Dual scattering is a method that approximates global illumination effects within the hair volume at a shading point $\mathbf{x}$ as a combination of two components: global scattering and local scattering, illustrated in Fig. 2.7 (a). The global scattering approximates how much light arrives at $\mathbf{x}$ after penetrating through n fibers along the light path. If the light comes from behind $\mathbf{x}$, it is directly seen by the camera, forming a *global scattered lobe*. If not, the arrived light will be added to direct illumination, going into the hair volume, scattering inside, then going back, forming a *local scattered lobe*.

Dual scattering begins with simplifying the complicated longitudinal lobes for each hair fiber as one forward lobe and one backward lobe. It first pre-computes the averaged forward/backward scattering intensity $\bar{a}_{f|b}$ as

$$\bar{a}_{f|b}(\theta_i) = \frac{1}{\pi}\iint_{\omega_{f|b}}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} S(\theta_i, \phi_i, \boldsymbol{\omega_r})\cos(\theta_r)\,\mathrm{d}\phi_i \mathrm{d}\boldsymbol{\omega_r}, \tag{2.18}$$

where $\boldsymbol{\omega_r} = (\theta_r, \phi_r)$ is the outgoing direction to the forward or backward hemisphere $\omega_{f|b}$.

Apart from the averaged forward/backward intensities, the forward/backward lobes also require their averaged longitudinal variances $\bar{\beta}^2_{f|b}$. Since the TT lobe is mostly forward and the TRT lobe is mostly backward, the variances directly take their squared roughness $\bar{\beta}^2_f = \beta^2_{TT}$ and $\bar{\beta}^2_b = \beta^2_{TRT}$. The R lobe is simply ignored.

To evaluate global and local scattered lobes, the key is to compute how the light attenuates and spreads longitudinally along a main path. Figures 2.7 (b) and (c) illustrate the computation. Dual scattering assumes that the azimuthal scattering is complicated, and it always becomes isotropic.

For the global scattered lobe, dual scattering calculates the attenuation of the light T_f reaching $\mathbf{x}$ after forward scattering through n fibers by taking the sequential product of their averaged forward attenuations. And the spread variance $\bar{\sigma}^2_f$ reaching $\mathbf{x}$ is computed by accumulating the pre-computed variances of all n fibers. Figure 2.7 (b) illustrates this idea.

The local scattered lobe is approximated similarly, as shown in Fig. 2.7 (c). The camera path goes into the hair volume, going forward through i hair fibers, scattering back once, then going forward through the previous i fibers until reaching the shading point $\mathbf{x}$ again. By summing up all possible paths with varying $i \geq 1$, the local attenuation with one backward scattering event A_1 can be computed. In order that the camera path finally returns to $\mathbf{x}$, backward scattering can also happen any odd number of times i.e. $3, 5, \ldots$. Dual scattering also computes A_3, and ignores higher-ordered scattering, so that the total attenuation for the backward scattered lobe is $A_b = A_1 + A_3$. The spread variance $\bar{\sigma}^2_b$ is the averaged accumulated variance along each possible path, weighted by its attenuation.

With global and local scattered lobes computed, the final approximate global illumination is the sum of both lobes, added to the hair BCSDF model. Specifically, the local scattering forms a lobe:

$$S_b(\theta_i, \theta_r, \phi) = \frac{I_b(\phi)}{\pi \cos^2 \theta_i} \cdot d_b A_b \cdot G(\theta_r + \theta_i; \bar{\sigma_b}^2), \tag{2.19}$$

where d_b is the backward scattering density constant, usually set between 0.6 and 0.8, while $G(\mu; \sigma^2)$ is a Gaussian with mean μ and variance σ^2. $I_b(\phi)$ is a binary backward hemisphere indicator which is 1 when $\phi \in [-\pi/2, \pi/2]$ and 0 elsewhere.

The global scattering forms another lobe:

$$S_f(\theta_i, \theta_r, \phi) = \frac{I_f(\phi)}{\cos^2 \theta_i} \cdot d_f A_f \cdot \sum_p \left(G(\theta_r + \theta_i; \bar{\beta}^2_p + \bar{\sigma_f}^2) N^f_p \right), \tag{2.20}$$

where d_f is the forward scattering density constant, usually set between 0.6 and 0.8. $I_f(\phi)$ is a binary forward hemisphere indicator which is 0 when $\phi \in [-\pi/2, \pi/2]$ and 1 elsewhere. $N^f_p = \frac{1}{\pi} \int_{\frac{\pi}{2}}^{\frac{\pi}{2}} N_p(\phi - \phi'; \eta') \, \mathrm{d}\phi'$ is the averaged azimuthal lobe p within the front hemisphere.

The dual scattering has made many assumptions, the most important of which is the main path assumption. In Chapter. 4, we demonstrate that this assumption is only reasonable for unscattered lobes (notations will be formally defined then). For scattered lobes, it will fail. So, in our model, we only use dual scattering as a component to handle paths consisting of unscattered lobes (TT and TRT, as in hair rendering).

2.6 Fourier Analysis and Sheared Filtering

As introduced in Chapter 1, speeding up ray tracing towards real-time require precise utilization of local smoothness of light transport, usually conducted by filtering as a post process to remove ray traced noise. This is independent to the reflectance of materials. In this subsection, we introduce the basic concepts, Fourier analysis and Sheared Filtering theory in flatland to motivate our sheared filtering for real-time ray traced realism. We will develop this concept in 3D and introduce the full 4D filter in Chapter 5.

We first introduce the general filtering configuration for soft shadows first. Let x denote receiver (surface visible at a pixel) coordinate and $y \in [-L, L]$ denote the light coordinate. Very similar parameterization and notation can be used for the lens coordinate for defocus, or incident direction parameterization for global illumination, and important details are mentioned below.

Soft shadows

Following Mehta et al. [78], we assume the light has a Gaussian intensity with standard deviation σ_y, and a side length $2L = 4\sigma_y$. For each pixel, we want to simultaneously filter and integrate light visibility and intensity, to compute the overall pixel irradiance. Let $f(x, y)$ be the visibility function and $I(y)$ be the Gaussian light intensity. Then the pixel irradiance is

$$h(x) = \int_{-L}^{L} f(x, y) I(y) dy. \tag{2.21}$$

It is shown in [31] that a single occluder plane at distance d_2 from the light, produces a single line of slope given by $s = d_1/d_2 - 1$ in the Fourier spectrum of f, when the receiver pixel is at a distance d_1 from the light source. With multiple occluders, most of the Fourier energy lies between lines of slopes $s_{\min}$ and $s_{\max}$, as shown in Fig. 2.8(a). These bounds can be estimated during the ray-tracing phase. The double-wedge spectrum of f is filtered by the light intensity spectrum on the ω_y axis, and this bandwidth is $\omega_y^{\max} = 4/L$. The computation of soft shadows theoretically requires that the receiver's material be diffuse, but in practice moderately glossy receivers also work, as shown by [78] and most algorithms based on shadow maps [43, 38, 2].

Depth of field

For rendering depth of field, $x \in [-W, W]$ is measured in pixel space, where W is the width of the image, and $u \in [-A, A]$ is on the lens, where $2A$ is the lens aperture.[1] The light field incident on the camera sensor in (x, u) space has a Fourier transform similar to the area-light visibility f. As shown in [80, 126], a plane at a single depth z produces a line of slope

[1] This normalization is chosen to be analogous to the soft shadow example, and is slightly different from Mehta et al. [80] who normalize the lens coordinate in $[-1, 1]$, and therefore have an extra aperture factor in their formula for circle of confusion.

$s = W(F/z - 1)/S$ in the fourier spectrum of the light field, which corresponds to the circle of confusion at that depth. Here F is the focal distance, and S is the size (meters) of the focal plane. Hence, most of the spectrum is bounded between the minimum and maximum circles of confusion, $s_{\min}$ and $s_{\max}$. The bandlimit due to the integration with the Gaussian lens aperture is $\omega_u^{\max} = 4/A$.

Diffuse indirect illumination

To get the double-wedge spectrum for the indirect light field, it must be parameterized in coordinates x along the receiver and v on a plane parallel to the receiver at unit distance. Then a single parallel reflecting surface at distance z from the receiver produces a line of slope $s = z$ in the light field spectrum in the (ω_x, ω_v) space. With multiple sloped reflectors, as shown in [79], we get a double wedge between slopes $s_{\min}, s_{\max}$. Finally, the double wedge is band-limited by the transfer function of the diffuse BRDF, given by:

$$\gamma(v_1, v_2) = \frac{1}{(1 + v_1^2 + v_2^2)^2}. \tag{2.22}$$

As derived in [79] the bandlimit $\omega_v^{\max} \approx 2.8$.

Sheared filtering

With the basic notations defined, we now introduce the *sheared filter* that our method build upon. Again, we illustrate it with its application on soft shadows. A similar formalism applies to depth of field and diffuse global illumination, except for a different choice of variables. As indicated in Fig. 2.8(a), the resulting Fourier-domain sheared filter has a shear slope given by the harmonic average of the min and max slopes, and the filter scale is proportional to the difference in the slopes [31]. We are only concerned with the primal domain filter, as shown in Fig. 2.8(b). The final filtered pixel irradiance $h(x)$ can be obtained as follows, using a 2D sheared filter w in flatland:

$$\begin{aligned} h(x) &= \iint f(x', y') w(x', y'; x, y)\, \mathrm{d}x'\, \mathrm{d}y' \\ &= \iint f(x', y') w_x(x' - x; \sigma_x) w_y(y' - y(x, x'); \sigma_y)\, \mathrm{d}x'\, \mathrm{d}y'. \end{aligned} \tag{2.23}$$

Both $w_x(\cdot), w_y(\cdot)$ are Gaussian functions, with standard deviations σ_x and σ_y respectively. σ_x depends on the sheared filter scale, and σ_y depends on the light bandlimit with:

$$\sigma_y = \frac{2}{\omega_y^{\max}} \qquad \sigma_x = \frac{2}{\omega_y^{\max}} \frac{s_{\min} s_{\max}}{s_{\max} - s_{\min}}. \tag{2.24}$$

The filter is a sheared spatially-varying convolution, with the center of the filter along the x' axis determined by the desired location x. The center of the filter along the y' axis is

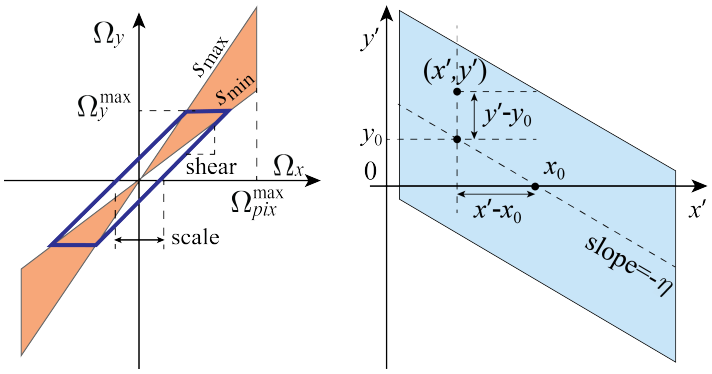


Figure 2.8: Illustration of the 2D sheared filter in flatland in (a) Fourier domain and (b) primal domain. The sheared filter in flatland gives the weight of a sample at (x', y') for a pixel of interest x. The filter can be split into two Gaussians: The x-axis Gaussian is fixed with center at x; the y-axis Gaussian has a varying center given by $y = \eta_x(x - x')$.

determined by the shear amount η_x, as in Fig 2.8(b),

$$\begin{aligned} y(x, x') &= \eta_x(x - x') \\ \eta_x &= -\frac{s_{\min} s_{\max}}{2(s_{\min} + s_{\max})} \end{aligned} \tag{2.25}$$

while the standard deviation σ_y remains constant and is related to the maximum bandlimit of the light, lens, or sloped reflectors in different applications. *Note that we have introduced the auxiliary variable y for the center of the filter along the y' axis.*

Note that, y is not an independent variable that immediately allows separation of the sheared filter. Also note that for different x, we have varying η_x values. This prevents us from separately integrating along the y'-axis, because the filter's center y is uncertain. However, in Chapter 5, we will prove that writing in this form finally leads an efficient factorization.

2.7 Summary

In this chapter, we have briefly introduced some background knowledge. We start from radiometry, then describe statistical BRDF, wave BRDF and basic light transport in preparation for our detailed rendering work in Chapter 3. Then we introduce different reflectance models for human hair fibers, together with an approximate model for global illumination between hair fibers. We will introduce the limitations of these methods for animal fur rendering and come up with our models in Chapter 4. Finally, we introduce the general Fourier analysis and sheared filtering related to our reconstruction scheme in Chapter 5. In the next sections, we will elaborate each of our research topics.

Chapter 3

Detailed Rendering from Complex Surfaces

3.1 Introduction

Conventional BRDFs model complex microgeometry using a smooth normal distribution function (NDF) of infinitely small microfacets. However, real surface features are certainly not infinitely small. Bumps and flakes from anywhere between a few microns (brushed metal) to about 0.1 mm (flakes in metallic paints) to centimeters (ocean waves) can produce interesting glinty behavior that is visible with the naked eye. These glints are very pronounced

Figure 3.1: A rendering of specular objects with extremely low roughness (standard deviation of 0.001 radians) under point illumination. A high resolution normal map (2048^2) with scratches and small-scale noise makes rendering impractical with standard Monte Carlo direct illumination, since the highlights are tiny and easily missed by naive pixel sampling. ***Left inset:*** *Our solution is based on the concept of a pixel normal distribution function ($\mathcal{P}$-NDF), which can be highly complex. Our algorithm evaluates it exactly, instead of using simple approximations.* ***Right inset:*** *Our method delivers an accurate solution, even in a temporal sequence with a moving light.*

with a light source that subtends a small solid angle, such as the sun and small light fixtures. This is true for surfaces specifically designed to glint, such as metallic paints with embedded flakes or decorative brushed metals, but also for everyday objects such as plastics or ceramics. In fact, smooth surfaces that meet the microfacet assumption are the exception rather than the norm. Most shiny surfaces that one encounters in reality have this type of glinty behavior, readily observed under sharp lighting.

Our goal is to simulate glinty appearance in still images and animations (Figure 3.1). Representing geometry at a resolution sufficient to reveal the features that cause glints is not difficult: we use high-resolution normal maps. A much harder challenge is rendering a complex specular surface under sharp lighting. Standard uniform pixel sampling techniques for direct illumination have extremely large variance, and using them for this purpose is impractical. The reason is that most of the energy is concentrated in tiny highlights that take up a minuscule fraction of a pixel, and uniform pixel sampling is ineffective at hitting the highlights (Figure 3.3). An alternative explanation is that the space of valid camera-surface-light paths is complicated and cannot be easily sampled from the camera or from the light. In some sense, we need to search the surface for normals aligned with the half vector, and this cannot be done by brute-force sampling.

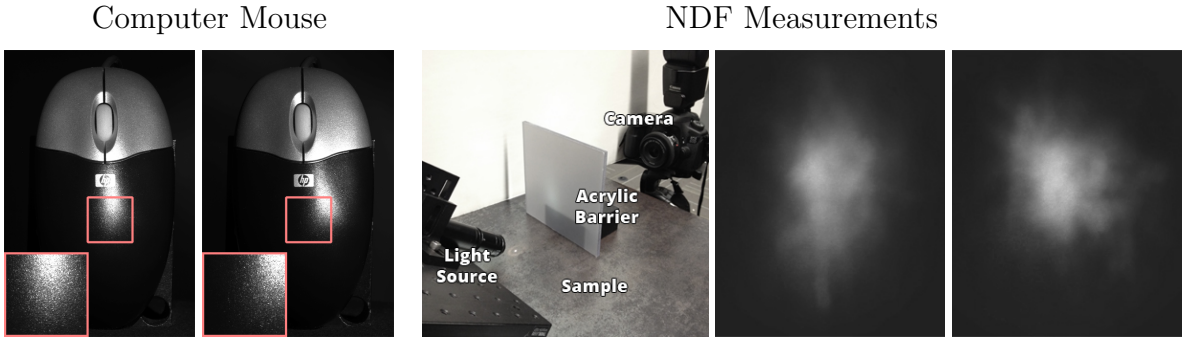


Figure 3.2: (a-b) Two photographs of an injection molded plastic computer mouse illuminated by a small light source ($\sim 3.5 \times 10^{-5}$sr solid angle) reveal its glinty appearance. These effects are impractical to simulate using uniform pixel sampling. (c-e) Real-world normal distribution functions of a dark bumpy ceramic tile were measured by illuminating the surface with a small focused incoherent source ($\sim 6.2 \times 10^{-5}$sr solid angle covering a surface patch of $\sim 0.5^2$mm). The images in (d) and (e) were captured by a camera located opposite a diffuse acrylic barrier from the source. They reveal a distinct non-Gaussian distribution of scattered light, corresponding to the $\mathcal{P}$-NDFof the surface patch, only slightly warped and blurred because of the optical limits of our setup.

Normal map filtering techniques [122, 40, 89, 28] also do not fully solve the problem. These methods attempt to approximate the NDF at a given scale by broad lobes, but the true NDF is highly complex; it cannot be approximated well using a single Gaussian lobe, or even a small number of lobes (Figure 3.4). Although these approaches avoid aliasing

artifacts, they are not able to reproduce glinty appearance under high-frequency lighting. We instead desire to compute the true solution that Monte Carlo would eventually converge to, using a completely different deterministic approach with minimal approximations.[1]

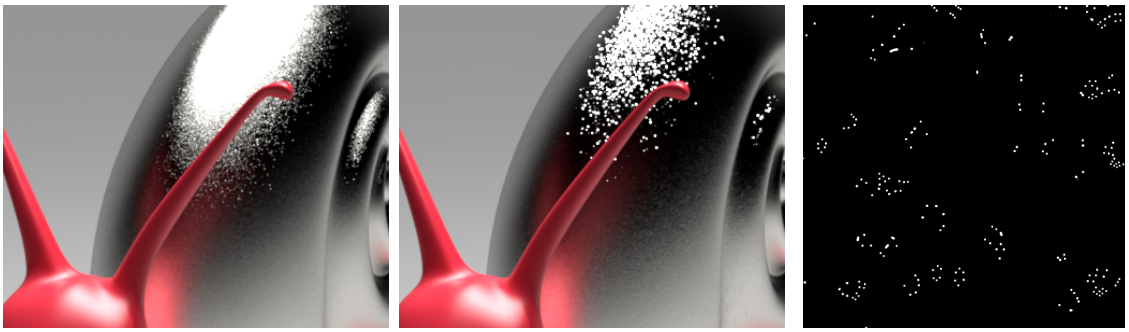
We consider the actual, unsimplified NDF of a surface patch $\mathcal{P}$ seen through a single pixel (an example is shown in Figure 1, left inset). This $\mathcal{P}$-NDF can be easily estimated by binning: repeatedly choose a point on the patch, take its normal, perturbed by the intrinsic surface roughness, and add it into a bin. The key problem is that for direct illumination, we need to evaluate the $\mathcal{P}$-NDF for a *single* half-vector. Clearly, it would be extremely inefficient to use the binning approach here, wasting all but a single bin. Indeed, this is equivalent to what a standard renderer would do, trying to hit a tiny light source by chance. Instead, we require evaluation of the density of a single normal coming from anywhere on the patch. Moreover, the $\mathcal{P}$-NDF is different for every pixel, so computations cannot be reused. In our method, the $\mathcal{P}$-NDFis just a mathematical tool to derive what the correct pixel brightness should be; it is never fully constructed, and only evaluated for a single vector.

We introduce an algorithm for $\mathcal{P}$-NDF evaluation in Section 3.4. The key assumptions that make the evaluation possible are a Gaussian pixel filter and a tiny amount of Gaussian roughness on the specular surface. These combine into a single 4-dimensional Gaussian "query" that is *analytically integrated* across the normal map, avoiding random sampling. A basic computational block of our solution is an integral of a 2-dimensional Gaussian over a triangular domain, described in Section 3.5. We hierarchically prune position-normal space to quickly find texels that might contribute to a given $\mathcal{P}$-NDFevaluation (Section 3.7). Our results show complex, temporally varying glints from bumpy plastics, brushed and scratched metals, metallic paint and ocean waves; see Section 3.7 and Figure 3.16.

In Section 3.8, we present an improved solution that exploits 4D Gaussian elements to speed up the $\mathcal{P}$-NDFcomputation by over 100 times, as compared to our basic triangular integration approach. This speed-up has significant benefits: it enables our method to be used as a standard BRDF in a Monte Carlo renderer, which is convenient from an engineering perspective. As a consequence, we can now use the BRDF with multiple importance sampling, and transparently handle all effects supported in Monte Carlo frameworks, such as illumination from environment maps and area lights. We provide details of implementing the Gaussian elements method in Section 3.9, and show speedups in Section 3.10 by comparing with our earlier method.

We further extend our method to handle wave optics. In Section 3.11, we present an algorithm that can evaluate a spatially varying BRDF for a given position and incoming/outgoing directions, by computing a wave optics reflection integral over the coherence areas around the position of interest. This requires more computation than in the geometric optics solutions, which makes our wave optics method slower, but not prohibitively so; see Table 3.3. Our solution is based on approximating the micron-resolution surface wave effects using Gabor kernels (products of Gaussians with complex exponentials). We use a reciprocal modification

[1]Specifically, constant view and light direction over $\mathcal{P}$, and the approximations made when solving the integral in Section 3.5.



Our method (17 min, 2.2 min actual glints) | Naive sampling (2 hours, 4,096 samples) | Zoom-in of a single pixel

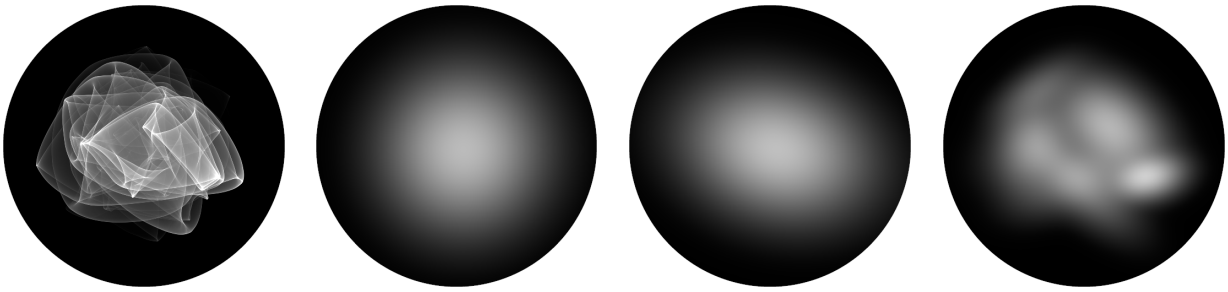
Figure 3.3: Naive pixel sampling fails at rendering complex specular surfaces under point lights. The reason is that the highlights are too small to be efficiently hit by uniform pixel sampling, which is obvious from the zoomed pixel on the right. Multiple importance sampling would not help since the light is a point, and it is the pixel integral that is inefficiently sampled, not the light/BRDF combination.

of the Harvey-Shack theory in our results, but our approach also applies to other wave optics models. We found that the difference between the geometric and wave solution is more dramatic when spatial detail is taken into account. The visualizations of the corresponding BRDF lobes differ dramatically, with the sharp folds typical of NDFs replaced by very different directional patterns more akin to laser speckle (Figure 3.23). The rendered highlights change appearance, typically with more realistic-looking sharper peaks and longer tails. Moreover, the wave optics solution varies as a function of wavelength, predicting noticeable color effects in the highlights (Figure 3.17). Our results show both single-wavelength and spectral solutions to reflection from common everyday objects, such as brushed, scratched and bumpy metals; see the result figures in Section 3.12.

3.2 Related Work

Naive pixel sampling. The standard approach to compute direct illumination on a bumpy specular surface is to trace a ray through the pixel, evaluate the normal of the hit point, and shade the point from a light source using the point's finite roughness BRDF; this fails at rendering glints (Figure 3.3). Multiple importance sampling [129] does not help, because it is the pixel integral that is inefficiently sampled, rather than the BRDF/light combination. The REYES approach of surface subdivision into micropolygons [12] is equally inefficient, since it would require micropolygons as small as the highlights. Though we use fine triangulations of the normal map for smoothness, our method can handle highlights that are arbitrarily smaller than the triangles.

Normal map filtering techniques can deliver artifact-free renderings by approximat-



correct NDF (our approach) | isotropic Gaussian [Toksvig 2005] | anisotropic Gaussian (LEAN, LEADR) | mixture of 10 aniso. [Han et al 2007]

Figure 3.4: Approximating the true NDF by a single Gaussian (Toksvig, LEAN, LEADR) or a small number of Gaussians (Han et al.) loses the sharp features that cause glinting.

ing a pixel's NDF by a single lobe [122, 89, 28] or a small number of lobes [40]. However, none of these methods can correctly capture glinty appearance. The core of the problem is that the true NDFs can be highly complex, and their sharp features directly translate into spatial and temporal glinting. Approximating them by broad lobes is only applicable under low-frequency illumination that would filter the complex features anyway. Figure 3.4 shows the effect of replacing the true NDF by a single Gaussian or a mixture of Gaussians, thus losing the sharp features.

Single-point evaluation of caustics. Caustics are related to our work, since glints can be interpreted as "directional caustics". Most methods sample paths (particles, photons) and accumulate them in a data structure (kd-tree, hash grid, or bins). However, this is not sufficient for our purposes; we require point evaluation, which is much harder. Walter et al. [133] compute volumetric caustics due to the refraction of a point light into a scattering volume through a bumpy interface. This is related to our approach: linear normal interpolation over triangles is used, a discrete set of specular connections is enumerated, the Jacobian determinant term determines highlight intensity, and a hierarchy is used to speed up the enumeration. However, no intrinsic roughness is considered (resulting in singularities), and the phenomenon rendered is quite different. Mitchell and Hanrahan [82] compute reflected caustics from an implicit surface by enumerating the discrete set of valid light paths through interval arithmetic. They used wavefront tracing as a way to compute the contribution of a valid specular path; this is again equivalent to the Jacobian determinant term for a single reflection, with the associated singularities.

Other work on specular paths. Jakob and Marschner [52] is an extension of Metropolis light transport, which allows mutation of a specular path at a single diffuse vertex; however, in our case, no diffuse vertex is available for mutations. In the perfectly specular case, there is a discrete set (rather than a manifold) of valid paths, as already noted above. Moon et al. published several approaches to approximate higher-order specular bounces, e.g. [84], but low-order specular paths are still computed brute-force with a relatively large light

source.

Stochastic reflectance. Jakob et al. [56] also addresses the problem of glinty surfaces, using a stochastic approach. Rather than work from a normal map, that method models the surface as a procedural random collection of specular flakes that occur according to a particular normal distribution. The key to their method is counting up the particles contributing to a particular illumination calculation without actually generating them, providing efficiency for large query areas where many particles contribute. When used as a model for a bumpy smooth surface, the stochastic approach is phenomenological: the random-flake approximation replaces the $\mathcal{P}$-NDF. In contrast, our algorithm exactly determines how a given specular surface, defined by a particular normal map, really looks under given sharp illumination. Moreover, normal maps can express surface features large enough to be visible in the image, e.g. the scratched and brushed examples in this dissertation.

Large area averaged wave optics. Rough surface reflection models based on wave optics have been heavily studied in physics. Common approximations include Beckmann-Kirchoff theory [4] and variations of Harvey-Shack theory[41]; a good overview is the thesis of Krywonos [64]. In graphics, wave-based reflection models have been developed for surfaces with stationary statistics, either random [45] or periodic [116], usually characterized by their power spectral density. A variety of methods have been proposed to measure such statistics for specific types of real surfaces, especially periodic ones [24, 121, 66]. Dong et al. [25] acquired the surface microgeometry of real metallic surfaces using a profilometer, and applied Kirchhoff theory to successfully predict their large scale BRDFs. A combined microfacet-diffraction model was recently proposed by Holzschuch and Pacanowski [48], demonstrating better fits to measured BRDF data for some materials than microfacet models alone. Levin et al. [71] designed special multi-planar surfaces that can be lithographically fabricated to match a target BRDF using wave optics, essentially inverting the rendering process.

Wave optics has also been used to predict appearance from thin-film or layered materials (e.g., [5]), but we will only consider single-layer opaque surfaces here. Several methods simulate longer-range multi-surface interference effects (e.g., [15]) but that is beyond the scope of this paper.

Spatially-varying wave optics. The only previous work we are aware of in this area is the recent paper by Werner et al. [138] (with a real-time extension by Velinov et al. [130]), rendering surfaces with collections of randomly oriented scratches using a Harvey-Shack-based wave optics model. This work represents the surface as a collection of one-dimensional scratches over a smooth BRDF. Under this assumption, they are able to compute the reflection efficiently and analytically. In contrast, our method can render arbitrary heightfields (e.g. Figure 3.26 and 3.27), including but not limited to ones containing scratches. Additionally, our scratched heightfields can contain more variety and imperfections, resulting in glinty highlights that only roughly align in lines, compared to the smooth line highlights of Werner et al (see Figure 3.17, esp. insets).

symbol	domain	definition
$\mathcal{D}$		unit disk (proj. hemisphere)
$\perp$		invalid normal
$\mathbf{s} = (s,t)$	$\mathcal{D}$	unit disk parameters, defining vectors $(s,t,\sqrt{1-s^2-t^2})$
$\mathbf{u} = (u,v)$	$\mathbb{R}^2$	texture space parameters
$n(\mathbf{u})$	$\mathbb{R}^2 \to \mathcal{D}$	normal map function
$J(\mathbf{u})$	$\mathbb{R}^2 \to \mathbb{R}^{2\times 2}$	Jacobian of $n(\mathbf{u})$
$\mathcal{P}$		pixel footprint
$G_p(\mathbf{u})$	$\mathbb{R}^2 \to \mathbb{R}$	pixel Gaussian
$G_r(\mathbf{s})$	$\mathbb{R}^2 \to \mathbb{R}$	intrinsic roughness Gaussian
$G_c[\mathcal{P},\mathbf{s}](\mathbf{x},\mathbf{y})$	$\mathbb{R}^4 \to \mathbb{R}$	combined Gaussian query for footprint $\mathcal{P}$ and normal $\mathbf{s}$
$D(\mathbf{s})$	$\mathcal{D} \cup \perp \to \mathbb{R}$	normal distribution function

Table 3.1: Notation used in the dissertation.

3.3 Preliminaries

Solving our problem requires thinking about a surface patch $\mathcal{P}$ seen through a pixel all at once, rather than one point at a time. Just as every surface point has a local BRDF, we can think of areas of the surface having $\mathcal{P}$-BRDFs that describe how the total contribution to the pixel depends on the illumination. Rendering detailed normal maps requires an efficient way to evaluate the area-integrated $\mathcal{P}$-BRDF, rather than letting the pixel filter do it implicitly by point sampling.

For a specular normal-mapped surface, this area-integrated BRDF is primarily determined by the distribution of surface normals over the relevant patch of surface: we need to be able to ask "how often" a given normal vector occurs in the patch. We call this distribution the $\mathcal{P}$-NDF; it is just like the microfacet distribution in a standard BRDF model, but it gives the normal distribution for a particular area rather than a global average over the whole surface. A crucial observation is that the $\mathcal{P}$-NDF is not a simple, broad function. It contains a surprising amount of structure (Figure 3.13) even when the surface patch is far larger than the features in the normal map. It also varies dramatically across the surface. Evaluating the $\mathcal{P}$-NDF efficiently while preserving this detailed spatio-angular structure is the key to accurately capturing glinty appearance.

Let us define these terms more precisely. Table 1 lists the symbols used throughout the dissertation.

Pixel footprint. We assume a Gaussian pixel reconstruction filter. This projects to an approximately Gaussian footprint $\mathcal{P}$ in the uv-parameterization of the normal map, whose covariance matrix is easily computed by propagating ray differentials to the surface [49]. In practice, we actually subdivide pixels into 4×4 subpixels, and make the footprints smaller accordingly. This handles edges better, but for simplicity we will talk about pixel rather

than subpixel footprints.

Projected hemisphere. We will use the unit disk $\mathcal{D}$ to express hemispherical unit vectors. The point $\mathbf{s} = (s,t) \in \mathcal{D}$ represents the unit vector $(s,t,\sqrt{1-s^2-t^2})$ on the hemisphere. Let us also define the *extended unit disk* as the union of the unit disk and a special symbol $\perp$, which allows for normal distributions that sometimes return invalid normals. This is less common than working with hemispheres, but it will be useful shortly.

Normal maps can be given directly or as the derivative of a heightfield. We use the direct option, though all but one normal map in our examples do come from a heightfield (the exception is the metallic paint flakes). The normal map is then defined as a function $n : \mathbb{R}^2 \to \mathcal{D}$ from points $\mathbf{u} = (u,v)$ in texture space to normals $\mathbf{s} = (s,t)$. The Jacobian of $n(\mathbf{u})$, denoted $J(\mathbf{u})$, plays an important role in determining highlight brightness, and points where $\det J(\mathbf{u}) = 0$ cause problems unless we are careful.

Intrinsic roughness. We could treat the surface as perfectly specular; however, we found that it is useful to consider a small amount of unresolved fine roughness. This matches the real world in that perfect smoothness is unachievable and the limits of geometric optics are reached at very high resolutions. It also prevents singularities (infinitely bright highlights), which arise with perfectly specular surfaces when $\det J(\mathbf{u}) = 0$, and cleanly deals with normal maps that contain piece-wise constant regions.

NDFs. We can now define a normal distribution function (NDF) as a probability distribution on the extended unit disk, with the obvious measure. (The associated random event is simply a "choice of normal".) This definition slightly deviates from standard references such as [132] and [7], but it is fully compatible with them, and is actually more convenient. In hemispherical terms, NDFs like Beckmann and GGX require an additional cosine term to integrate to 1, and their associated sampling routines also bake in a cosine (see eq. (4) and (28) in Walter et al. [132]); in our formulation, no cosines need to be worried about. Furthermore, we now have more freedom in what passes as an NDF: any suitable plane function can be restricted to the unit disk and properly normalized. In particular, Gaussians are perfectly good NDFs, and this includes anisotropic and non-centered ones. Finally, statements such as "blur an NDF by a Gaussian" now have a very precise meaning. Even though this is different from spherical convolutions with vMF or Kent distributions, the difference is not critical to us: we simply use the convolutions to avoid singularities coming from unrealistically perfect surfaces.

The $\mathcal{P}$-NDF can now be defined as the probability distribution of the random variable defined by sampling the footprint $\mathcal{P}$, evaluating the normal at the sampled location, and perturbing by the intrinsic roughness kernel. The last step can sometimes result in a normal outside of the unit disk; these events are collected by the probability of $\perp$, and are often near zero in practice. Figure 3.13 shows different $\mathcal{P}$-NDFs as the size of the pixel footprint increases. Note that quite large footprint sizes are required for these NDFs to start to mimic analytic normal distributions like Beckmann.

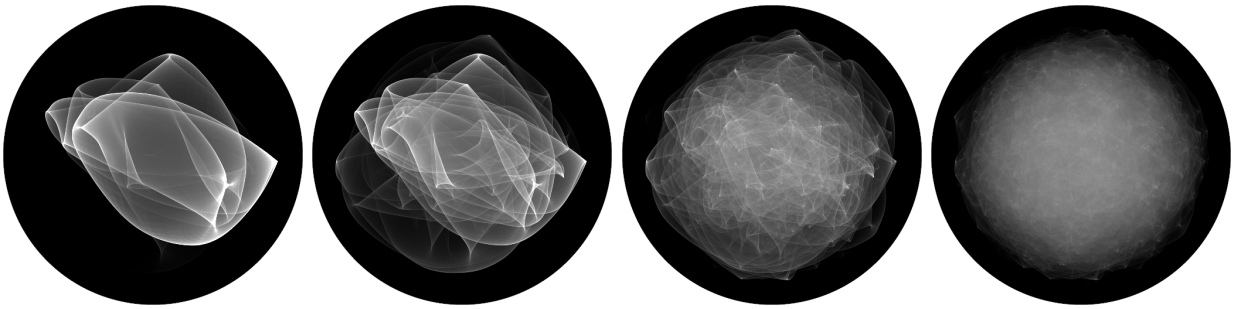


Figure 3.5: The $\mathcal{P}$-NDFs of a smooth specular heightfield with a Gaussian power spectrum, with a pixel footprint covering about 15×15, 30×30, 90×90 and 300×300 texels respectively.

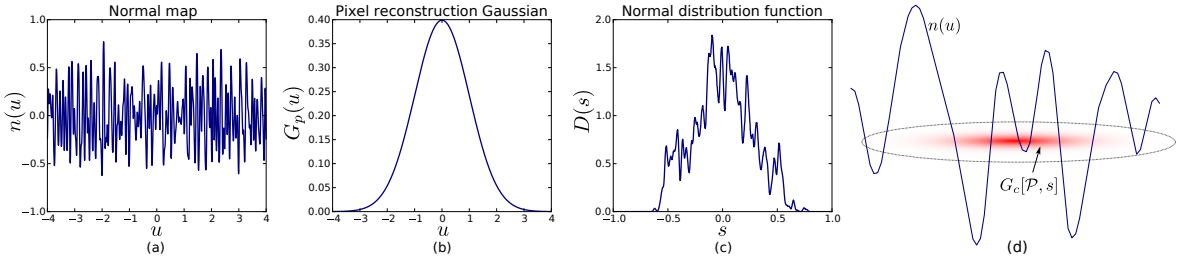


*Figure 3.6: Flatland illustration of $\mathcal{P}$-NDFsampling and evaluation. **(a)** A normal map is a 1D curve $n(u)$ of the texture coordinate u. (The other component of the normal vector is $\sqrt{1-n(u)^2}$). **(b)** The pixel of interest projects to a Gaussian footprint given by $G_p(u)$. **(c)** The $\mathcal{P}$-NDF$D(s)$ giving the probability density of a given normal $(s, \sqrt{1-s^2})$, assuming an intrisic roughness kernel $G_r(s)$ with $\sigma = 0.01$. **(d)** $\mathcal{P}$-NDFevaluation in flatland can be visualized as integration of the combined Gaussian query $G_c[\mathcal{P}, s]$ over the segmented graph of the normal map. In areas where the Gaussian is effectively zero (outside of the ellipse) we can prune the segments using a hierarchy.*

3.4 $\mathcal{P}$-NDFEvaluation in Flatland and 3D

Our core challenge is to find an evaluation algorithm for the $\mathcal{P}$-NDF$D(\mathbf{s})$ for a half-vector $\mathbf{s}$, corresponding to a given footprint on a given normal map and with a given intrinsic roughness; indeed, with such an algorithm at hand, it is straightforward to plug the $\mathcal{P}$-NDFinto a standard microfacet BRDF, which can be used for direct illumination calculations:

$$f_r(\mathbf{i}, \mathbf{o}) = \frac{F(\mathbf{i}.\mathbf{h})G(\mathbf{i}.\mathbf{h})D(\mathbf{h})}{4\ (\mathbf{i}.\mathbf{n})\ (\mathbf{o}.\mathbf{n})} \tag{3.1}$$

where $\mathbf{h} = (\mathbf{i} + \mathbf{o})/\|\mathbf{i} + \mathbf{o}\|$ is the half vector, $\mathbf{n}$ is the unmapped surface normal, F is the Fresnel term, and G is a shadowing-masking term (only needed to avoid infinities at grazing). In the following sections, we will first make the $\mathcal{P}$-NDFevaluation problem more approachable by analyzing the situation in flatland, and then present the full 3D solution, which naturally follows from the flatland case.

The flatland situation is simpler: there is only one texture parameter u. The normal

map can be written as a function $n(u)$ returning normals in $(-1, 1)$, which is analogous to the unit disk from the 3D case. The full normal vector is $(n(u), \sqrt{1 - n(u)^2})$. The pixel footprint $\mathcal{P}$ will turn into a Gaussian reconstruction kernel $G_p(u)$ that integrates to 1. Let X be a random variable that is distributed according to $G_p(u)$. The key question is, what is the distribution of the random variable $n(X)$ on $(-1, 1)$? This is not a simple multiplication or convolution of the normal map with G_p, but instead a pdf of a dependent random variable. The situation is illustrated in Figure 3.6.

We can write down the $\mathcal{P}$-NDFas:

$$D(s) = \int_{-\infty}^{\infty} G_p(u)\delta(n(u) - s)du = \sum_i \frac{G_p(u_i)}{|n'(u_i)|}, \tag{3.2}$$

where u_i are the roots of the equation $n(u) = s$. The delta function restricts the integral to points where $n(u) = s$, and the second equation intuitively accounts for the "speed" of crossing the root; it only works if a finite set of roots exists. As we can see, the $\mathcal{P}$-NDFwill have singularities at points where $n'(u) = 0$. These correspond to inflection points of the original heightfield. This analysis shows that the $\mathcal{P}$-NDFcan have infinite values. If we use a pinhole camera and a point light, this can cause infinitely bright pixels. (Our distant light/camera approximation is not the culprit; infinities could occur even if we did not make this approximation.) Furthermore, there could be constant regions in the normal map, so we get $n'(u) = 0$ for whole intervals, and corresponding delta functions in the $\mathcal{P}$-NDF.

To avoid singularities and other problems inherent in perfect specular surfaces, we introduce a tiny amount of finite roughness to the normal-mapped surface. Since the $\mathcal{P}$-NDFis just a function on the interval $(-1, 1)$, we can convolve it with a Gaussian $G_r(s)$ easily:

$$\begin{aligned} D(s) &= \int_{-1}^{1} G_r(s - s') \int_{-\infty}^{\infty} G_p(u)\delta(n(u) - s')duds' \\ &= \int_{-\infty}^{\infty} G_p(u) \int_{-1}^{1} G_r(s - s')\delta(n(u) - s')ds'du \\ &= \int_{-\infty}^{\infty} G_p(u)G_r(n(u) - s)du \\ &= \int_{-\infty}^{\infty} G_c[\mathcal{P}, s](u, n(u))du. \end{aligned} \tag{3.3}$$

In the last step, we combined the two 1D Gaussians into a single 2D one:

$$G_c[\mathcal{P}, s](x, y) = G_p(x)G_r(y - s). \tag{3.4}$$

By changing the integration order and eliminating delta functions, we have removed any notion of root finding or singularities from the problem, leaving a single well-defined integral of a one-dimensional real function. An elegant way to intuitively visualize the result is that we would like to integrate the combined reconstruction kernel $G_c[\mathcal{P}, s]$ along the *graph* of the

normal function, the plane curve $(u, n(u))$. Note, though, that the measure is the standard line measure on the u axis, not arc length along the graph. Figure 3.6 (d) illustrates this intuition, and immediately leads to an accelerated query idea: we can use a hierarchy to prune all normal map segments in areas where $G_c[\mathcal{P}, s]$ is effectively zero.

In flatland, G_c is a 2D Gaussian, so we can subdivide the graph into many line segments, and integrate the combined kernel along the line segments. This leads to integrals of 1-dimensional Gaussians over the segments, which can be computed easily in terms of $\mathrm{erf}(\cdot)$. This shows the benefit of choosing Gaussian filters; other choices such as splines would lead to integration problems without closed-form solutions.

Also note that we made the *graph* piecewise-linear, instead of the full integrand $G_c(u, n(u))$: the latter would be a bad choice, since *the Gaussian can be much narrower than the discretization step*. We would like to handle specular highlights arbitrarily smaller than the finest discretization level, and this choice is key to achieving that goal.

3D analysis. We can extend the above line of thinking to three dimensions, with two-dimensional texture space parameterized by $\mathbf{u} = (u, v)$, and a normal function $n : \mathbb{R}^2 \to \mathcal{D}$.

A 2D Gaussian reconstruction kernel $G_p : \mathbb{R}^2 \to \mathbb{R}$ now models the pixel footprint $\mathcal{P}$. The random process of choosing a position $\mathbf{u}$ by sampling G_p and taking its normal will have the following probability distribution:

$$D(\mathbf{s}) = \int_{\mathbb{R}^2} G_p(\mathbf{u})\delta(n(\mathbf{u}) - \mathbf{s})d\mathbf{u} = \sum_i \frac{G_p(\mathbf{u}_i)}{|\det J(\mathbf{u}_i)|}. \tag{3.5}$$

This is in direct analogy to the flatland derivation. While the flatland case has singularities at the inflection points of the original one-dimensional heightfield, here we have singularities at $\det J(\mathbf{u}) = 0$, which is a set of curves in uv-space where the curvature of the original heightfield flips between elliptic and hyperbolic. These curves directly correspond to the "folds" we often see in $\mathcal{P}$-NDFvisualizations. Again, piecewise constant normal maps (or affine regions of the heightfield) make $\det J(\mathbf{u}) = 0$ over whole regions, causing delta functions in $D(\mathbf{s})$. In fact, we have tried to implement eq. (3.5) using analytic root finding and found it impractical due to the singularities.

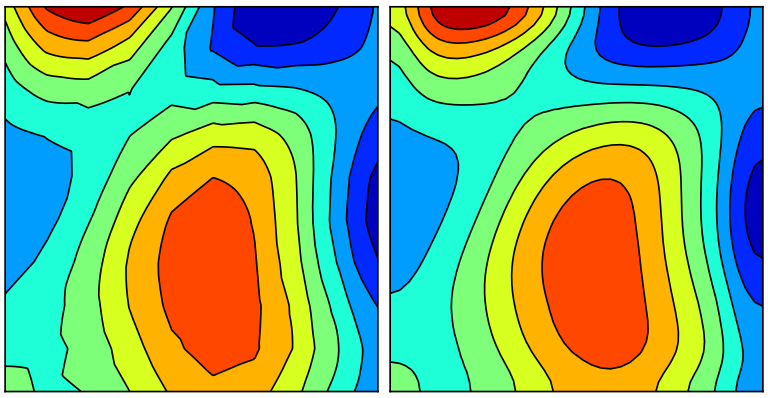
Therefore, as in flatland, we introduce intrinsic roughness. This is accomplished by a 2-dimensional Gaussian kernel $G_r(\mathbf{s})$, which convolves the $\mathcal{P}$-NDF. The derivation is identical to flatland except with bold letters:

$$\begin{aligned} D(\mathbf{s}) &= \int_{\mathcal{D}} G_r(\mathbf{s} - \mathbf{s}') \int_{\mathbb{R}^2} G_p(\mathbf{u})\delta(n(\mathbf{u}) - \mathbf{s}')d\mathbf{u}d\mathbf{s}' \\ &= \int_{\mathbb{R}^2} G_c[\mathcal{P}, \mathbf{s}](\mathbf{u}, n(\mathbf{u}))d\mathbf{u}. \end{aligned} \tag{3.6}$$

where

$$G_c[\mathcal{P}, \mathbf{s}](\mathbf{x}, \mathbf{y}) = G_p(\mathbf{x})G_r(\mathbf{y} - \mathbf{s}) \tag{3.7}$$

We can again visualize this intuitively as integration of the combined 4D reconstruction kernel $G_c[\mathcal{P}, \mathbf{s}]$ along the graph of the normal function, $(\mathbf{u}, n(\mathbf{u}))$, which is a 2D surface in



2 triangles / texel 32 triangles / texel

Figure 3.7: A patch of the normal map with 9 × 9 texels. The z-component of the normal is visualized using iso-lines, to clearly depict curvature discontinuities. Using 32 triangles per texel shows better smoothness than 2, at no extra storage.

4D space. This is hard to plot; however, the intuition that the graph can be triangulated and G_c reduces to 2D Gaussians over the triangles is correct. The hierarchical pruning idea also carries over from flatland.

In summary, we have observed that the $\mathcal{P}$-NDF $D(\mathbf{s})$ is not trivially evaluated at a single point (direction) $\mathbf{s}$. However, under Gaussian pixel and roughness kernels, we have cast this evaluation as an integration problem, which can be solved by discretizing the normal map into small affine patches. (Note, though, that the specular highlights we handle can still be much smaller than the patches.) The next section discusses the details of solving this integration problem.

3.5 Analytic integration

To numerically evaluate equation (3.6), we choose to discretize the normal map $n(u, v)$ into triangles, and linearly interpolate the normal across them. More precisely, we linearly interpolate the s and t values; the third coordinate is implied.

The simplest solution is to split each normal map texel into two triangles. This is sometimes sufficient, but we found that this discretization can produce triangular artifacts in the $\mathcal{P}$-NDF, if the resolution of the normal map is too low compared to the features it depicts. If this is an issue, we can up-sample the normal map, or subdivide texels into 4×4 sub-texels using bicubic Catmull-Rom interpolation. Any other subdivision could be used, but 4×4 naturally matches the control polygon of the bicubic patch. Figure 3.7 shows the difference between the two options.

Integrating a 2D Gaussian over a triangle $\triangle$. Our goal is to compute integrals of

the form

$$I = \int_{\triangle} G_c(\mathbf{u}, n(\mathbf{u}))d\mathbf{u} = \int_{\triangle} G(\mathbf{u})d\mathbf{u}. \tag{3.8}$$

Since we linearly interpolate the normals, n is an affine function on $\triangle$, which allows us to collapse the 4-dimensional combined Gaussian G_c into some other 2D Gaussian G.

This problem has been studied, and an R package implements one possible solution [95]. There exist numerical algorithms for evaluating the cumulative distribution function $\Phi(x, y, \rho)$ of a bivariate Gaussian with $\sigma_x = \sigma_y = 1$ and covariance ρ [35], which can be adapted to evaluate the desired integral. The PolyCub package also takes a similar approach. We have implemented this method and it works correctly, but appears slower than our method. A related problem for spherical Gaussians has been studied by Xu et al. [139].

Below we describe the implementation that we found to perform well in our case. $\triangle$ is a triangle from our triangulation; due to its construction, we only have right triangles, with two sides aligned to the axes. If $\triangle$ is the triangle given by $(u_0, v_0), (u_1, v_0)$ and (u_0, v_1), we obtain an integral

$$I = \int_{u_0}^{u_1} \left(\int_{v_0}^{f(u)} G(u, v)dv \right) du, \tag{3.9}$$

where $f(u)$ achieves a triangular integration domain:

$$f(u) = \frac{(u_1 - u)v_1 + (u - u_0)v_0}{u_1 - u_0}. \tag{3.10}$$

So far, we have just explicitly stated the problem. Eliminating v by carrying out the inner integration, and substituting x for the argument of the resulting erf function, this leads to integrals of the form

$$\text{experf}(a, b, x_0, x_1) = \int_{x_0}^{x_1} \exp(-a(x - b)^2)\text{erf}(x)dx \tag{3.11}$$

for some constants a and b, and shifted bounds x_0 and x_1. In fact, the same form will result if we center the triangle instead of the Gaussian, or if we transform the problem so the Gaussian is unit, or with any other similar approach. This integral does not have an elementary solution, but we can approximate it as follows.

We choose to approximate the function $\text{erf}(x)$ on the interval $[-3, 3]$ by a piece-wise quadratic function on six subintervals, and as -1 and 1 for $|x| \geq 3$. The problem thus separates into integrals of the form

$$\text{expquad}(a, b, c_0, c_1, c_2, x_0, x_1) = \int_{x_0}^{x_1} \exp(-a(x - b)^2)(c_0 + c_1 x + c_2 x^2)dx, \tag{3.12}$$

which can be solved analytically using a computer algebra system. The result is long but not fundamentally difficult.

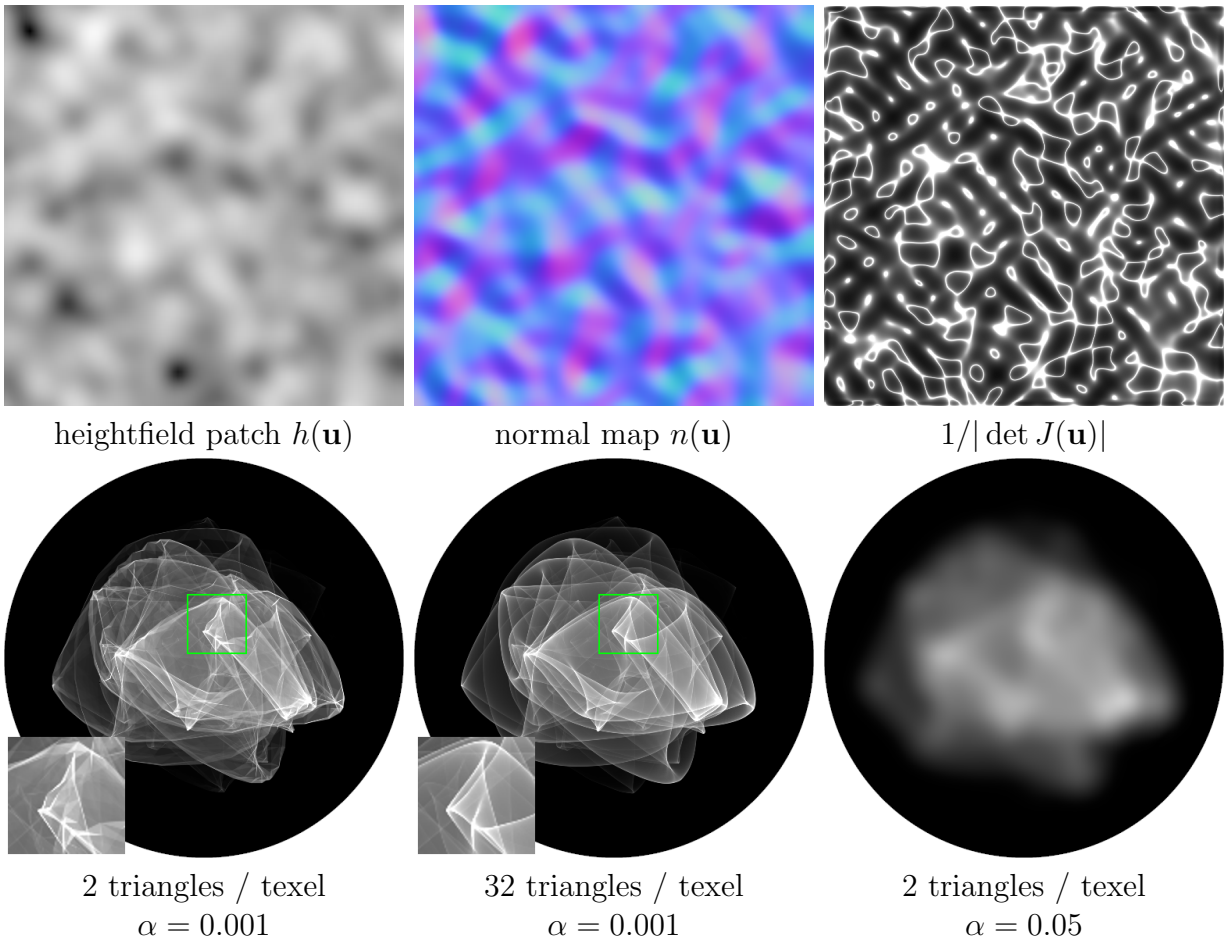
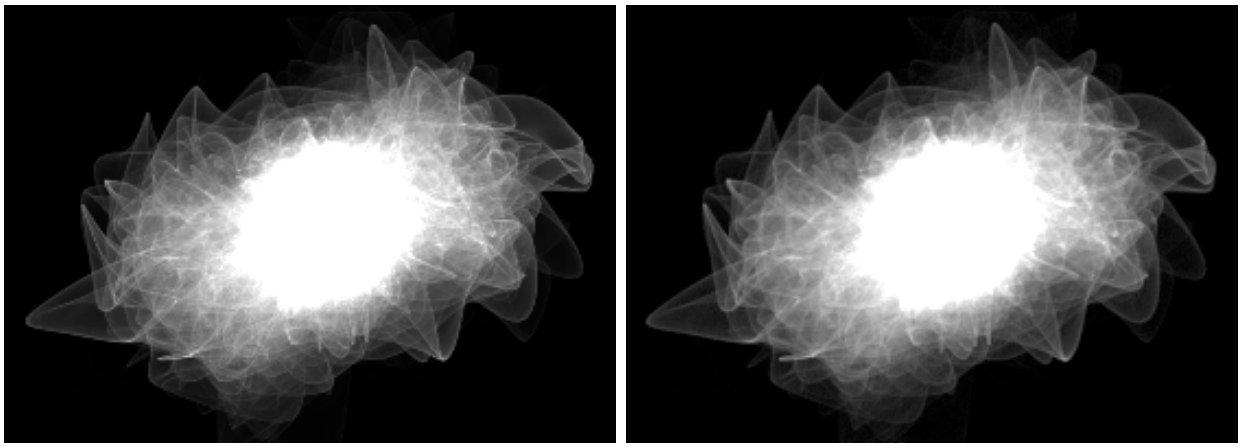


Figure 3.8: ***Top row:*** *A heightfield $h(\mathbf{u})$ with a Gaussian power spectrum, its normal map $n(\mathbf{u})$ and the $1/|\det J(\mathbf{u})|$ term that specifies the highlight brightness on a perfectly specular surface (with singularities at points where the original heightfield flips curvature).* ***Bottom row:*** *the $\mathcal{P}$-NDFcorresponding to the footprint, computed using our approach. Left to right, with roughness 0.001 and two triangles per texel (showing some artifacts), with 32 triangles per texel, and with roughness 0.05 and 2 triangles per texel (no artifacts).*

Figure 3.8 illustrates the result of our integration algorithm on a particular normal map patch.

Comparison against reference. The correctness of the derivation can be easily checked against the binning method. That is, we use 100 million samples of G_p, look-up the normal map, perturb by G_r, and store the samples in bins. Figure 3.24 shows the result. The time-sequence comparison in Figure 1 is also computed using this method. Note the excellent match between the two images, computed using completely different methods. A minor difference comes from the fact that the binning inherently computes bin integrals instead of bin center values like our evaluation. The supplementary data contains several different



our evaluation
32 triangles/texel

binning
100 million samples

Figure 3.9: Comparison of the $\mathcal{P}$-NDFevaluated by our approach to the reference $\mathcal{P}$-NDFcomputed by binning, demonstrating the correctness of our derivations, for a single pixel of the cutlery model. A minor difference comes from the "anti-aliasing" of the binning method, which naturally computes bin integrals instead of bin center values like our evaluation.

NDFs compared against the reference, in floating point format. Note that we only provide single-pixel rather than full-frame reference comparisons, since the latter would be extremely slow to compute using the 100 million samples (see Figure 3.3), and would arguably provide less insight than NDF comparisons.

3.6 Implementation

Hierarchical pruning of texels. To increase performance, we limit the Gaussians to be non-zero only within 5σ (a reasonable approximation). Therefore, many texels can be pruned, because either G_p or G_r are zero over the whole texel. We can trivially reject texels that fall outside of G_p. For G_r we utilize a min-max hierarchy over the normal map. More precisely, for each texel, we precompute the minimum and maximum value of $s(u, v)$ and $t(u, v)$, and build a quad-tree hierarchy over these bounds. For a given query of $D(\mathbf{s})$, we traverse the hierarchy, pruning whole groups of texels where G_r is guaranteed to be beyond 5σ. The recursive traversal is similar to many other bounding volume approaches.

Importance sampling. Sampling from a $\mathcal{P}$-NDFis easy by definition, using the same technique as was used to create the binning reference: simply take the normal of a random surface point seen through the pixel, and perturb by the intrinsic roughness kernel.

Adding other light paths. In our implementation, we separate the glint component of the image (i.e. direct illumination on normal-mapped specular surfaces from point lights) from all other light paths, which are computed using path tracing; any other standard algorithm could be used as well. On the first bounce from the camera, we use the full normal

map for importance sampling. On further bounces we use a global $\mathcal{P}$-NDFapproximation for both sampling and evaluation, since an accurate $\mathcal{P}$-NDFno longer makes a difference here. We could also use a normal map mip-mapping method in that case. A simple extension would be to smoothly transition to a normal map mip-mapping method in the distance, once glinting becomes insignificant.

Alternatively, our algorithm can be treated as a new "black-box" BRDF with an additional pixel footprint specification, while keeping all other parts of a renderer unmodified. However, we prefer to get separate timings, and we wanted to make sure the glint component is completely deterministic, to avoid any confusion about how much noise comes from the true glints vs. the algorithm. For this reason, we also do not use area lights, depth of field, or motion blur in our results, though they would be easy to add.

3.7 Results

Our implementation uses the Mitsuba framework [51], and runs on a 6-core Intel i7-4770K desktop at 3.5 GHz, hyperthreaded to 12 threads. Below we describe the scenes shown in Figure 3.16. Please see their temporal versions in the attached video. Note how the strong glinting is correct, given the normal map and the lighting; our method is entirely deterministic and does not produce any Monte Carlo noise. Our timings (Table 3.2) refer to one frame (1280×720). Note how the overhead of our algorithm is smaller than the standard rendering with other light paths. Also note that our performance depends on the number of pixels with glinty materials, and is independent of scene complexity.

Snail. This scene illustrates, on the snail's shell, a smooth heightfield created by inverse FFT from an isotropic Gaussian spectrum with randomized phase, converted to a normal map. The features of the normal map are smaller than a pixel, and yet the result is far from smooth, producing a fairly dramatic glint effect.

Metallic paint snail. Metallic paint, often used on cars, is specifically designed to show glints. Composed of several layers, the most important are the top clear-coat (which provides the smooth specular highlight) and the colored absorptive layer with embedded aluminum flakes [104]. We model the flakes using a normal map that is constructed by clustering the pixels into Voronoi cells, whose centers are chosen using Poisson disk sampling, and assigning a fixed normal to each cell, drawn from the Beckmann distribution. No normal interpolation is necessary (or desirable) in this case: each texel has a constant normal. No subdivision beyond 2 triangles is required either. We also added a diffuse lobe to approximate multiple internal reflections between the flakes and the clear-coat. The snail is about 10 cm long, making the flakes more visible than on a car.

Blender. This scene shows an energy drink blender with a bumpy plastic body and a brushed metal lid. Brushed metal is notoriously difficult to render under sharp lighting; typical compromises include increasing groove size, light size and roughness to unrealistic levels. None of this is necessary with our approach. We generated a normal map using the inverse FFT approach but with an anisotropic Gaussian power spectrum, and added noise

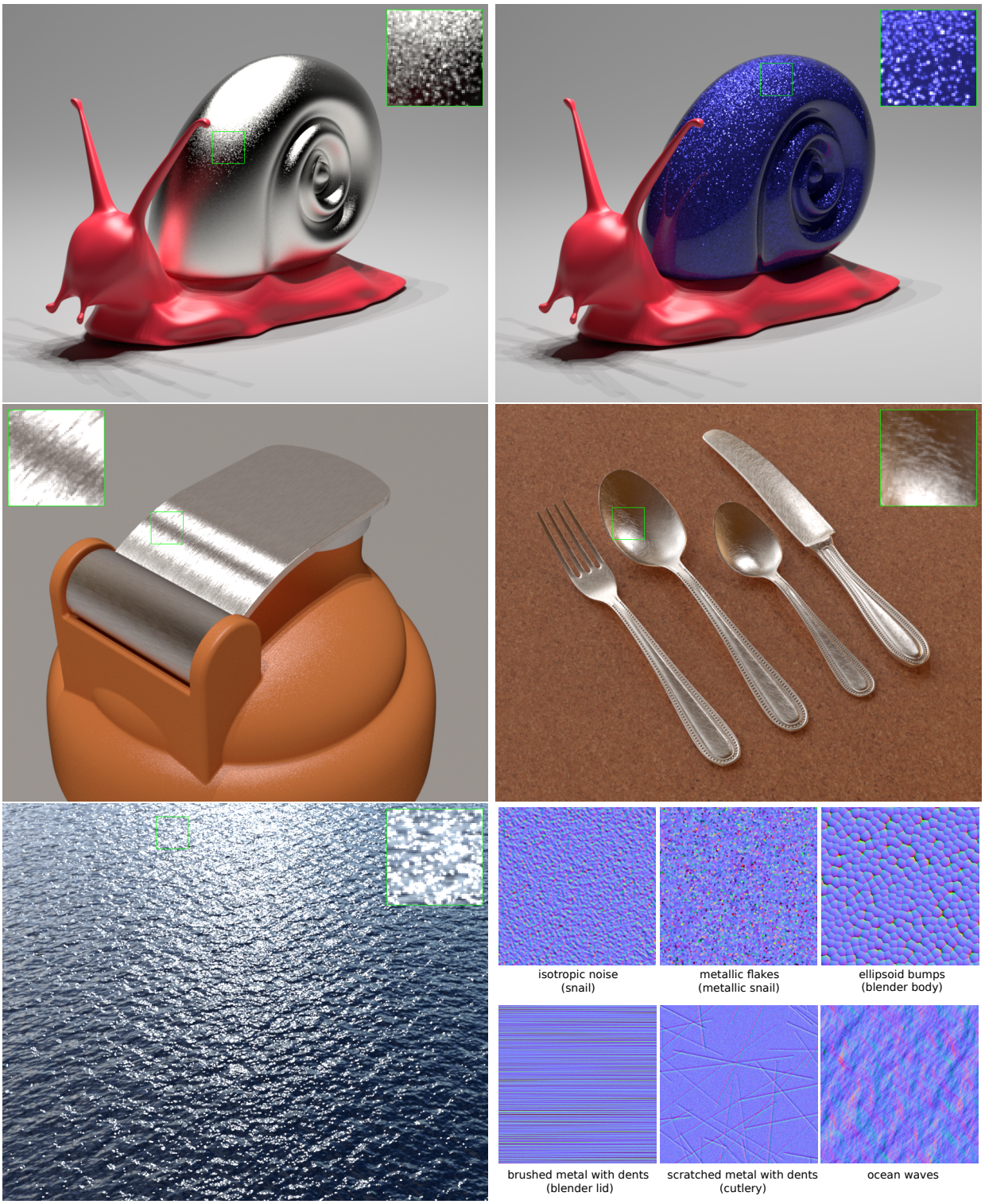


Figure 3.10: Still frames from our five scenes: snail (showing a simple isotropic noise normal map), metallic paint snail (modeling metallic flakes embedded in paint), blender (showing brushed metal with dents and plastic with ellipsoid bumps), cutlery (scratched metal with dents) and ocean (temporally varying waves caused by wind). We used simple sRGB in these images, but any tone-mapping could be applied. The full animations are shown in the supplementary video. Normal map contrast was enhanced for visualization purposes.

	Snail	Metallic	Blender	Cutlery	Ocean
Our	**2.2**	**1.0**	**5.5**	**6.2**	**9.9**
Global	15.6	19.5	19.0	8.7	-
Envmap	-	-	20.9	6.1	23.5
Total	17.8	20.5	45.4	21.0	33.4

Table 3.2: Timings of a typical frame in minutes on a 6-core hyperthreaded i7 machine. "Our" refers to the runtime of our direct illumination algorithm, the rest is the cost of standard path tracing. We split environment lighting into a separate component.

to the normals to simulate tiny dents. For the blender body, we used an ellipsoid bump heightfield, which produces glints of different appearance from the snail.

Cutlery. This scene shows metallic cutlery with strong scratches from heavy use. A configuration like this, under strong small LED lighting fixtures, is often seen in restaurants. We generated the scratches as randomly oriented, slightly blurred line-shaped valleys. We then added dents through noise, like with brushed metal above.

Ocean waves. Finally, we show our method applied to the ocean, with similar but larger features than previous examples. Here we model the ocean as a single rectangle with a normal map generated using the inverse FFT method [120]. While good anti-aliased ocean renderings have been possible using LEAN or LEADR methods, we can produce very sharp and correct glints even in the distance, where multiple waves project to a pixel.

3.8 An Improvement using Gaussian Elements

As demonstrated in previous sections, our basic triangulation-based solution is able to generate detailed glints from complex surfaces. However, this approach is still expensive and lack full generality in material and illumination support. In this section, we introduce an efficient and general method that can be easily integrated in a standard rendering system.

A key design decision of our improved approach is the representation of the high-resolution specular surface detail. We propose to represent the normals of the surface, with intrinsic roughness applied, as a 4-dimensional *position-normal distribution*. This distribution can be accurately approximated as a mixture of millions of 4-dimensional Gaussians (on the order of the number of texels). We will denote these Gaussians as *elements*.

Below, we first explain this idea in flatland (where the Gaussians are 2-dimensional), and the next subsections will transfer our intuitions to the full 4D case.

Mathematical framework in flatland

In flatland, the domain analogous to the unit disk (projected hemisphere) is simply the interval $[-1, 1]$. We repeat the $\mathcal{P}$-NDFevaluation in flatland with intrinsic roughness (Equa-

tion 3.3) as:

$$\begin{aligned} D_{\mathcal{P}}(s) &= \int_{-1}^{1} G_r(s-s') \int_{-\infty}^{\infty} G_p(u)\delta(n(u)-s')\,\mathrm{d}u\,\mathrm{d}s' \\ &= \int_{-\infty}^{\infty} G_p(u)G_r(n(u)-s)\,\mathrm{d}u. \end{aligned} \tag{3.13}$$

Earlier, we evaluate the integral by assuming $n(u)$ is piecewise linear, which works but leads to harder integrals without closed form solutions. We instead precompute an approximation to the whole second factor, $G_r(n(u)-s)$. This function is what we propose to be our surface representation, the position-normal distribution $\mathcal{N}(u,s)$. Note that both G_r and G_p are normalized (they integrate to 1). The top row of Figure 2.8 also demonstrates the idea: (a) shows a flatland surface and (b) the position-normal distribution of this surface. Note that normalization of $\mathcal{N}$ over u is not needed; it is sufficient that G_r is normalized over s.

Next, we propose to approximate $\mathcal{N}$ as a sum of 2D Gaussians in u and s, with scaling coefficients c_i, means $\mathbf{x}_i$ and covariance matrices $\mathbf{\Sigma}_i^{-1}$:

$$\mathcal{N}(u,s) = G_r(n(u)-s) \approx \sum_{i=1}^{m} G_i(u,s), \tag{3.14}$$

where

$$G_i(u,s) = c_i \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^T\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{x}_i)\right) \tag{3.15}$$

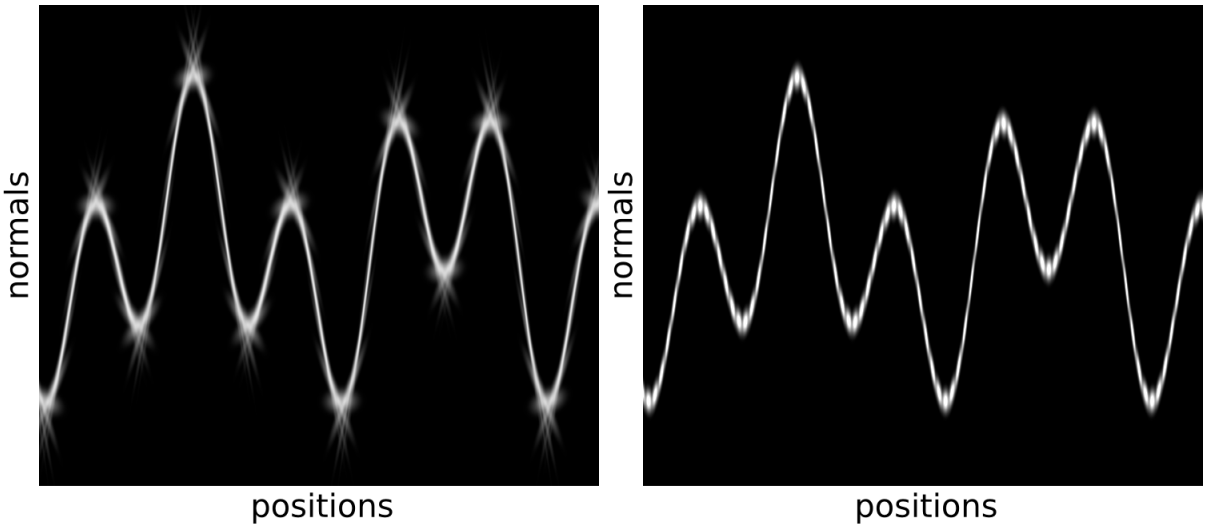
and $\mathbf{x} = (u,s)^T$. A key advantage of this representation is that the $\mathcal{P}$-NDFquery (for a given $\mathcal{P}$ and s) can be written as:

$$D_{\mathcal{P}}(s) = \int_{-\infty}^{\infty} G_p(u)\mathcal{N}(u,s)\,\mathrm{d}u \approx \sum_{i=1}^{m} \int_{-\infty}^{\infty} G_p(u)G_i(u,s)\,\mathrm{d}u. \tag{3.16}$$

Conveniently, the integral inside the sum has an efficient closed-form solution: as s is fixed, this is a product of two 1D Gaussians, which is just another Gaussian, easily integrable on the infinite domain. Fortunately, this convenience will carry over to the full 4D case. This is in contrast to our previous approach, which leads to much harder finite-domain integrals.

Creating a Gaussian mixture

A key question is how to create the Gaussian mixture approximation to $\mathcal{N}(u,s)$. It would be interesting to consider hierarchical EM techniques [131, 53], but typical applications of EM start from discrete samples. In our problem, the distribution is available explicitly. Specifically, we have access to the Jacobian of the normal map, letting us tightly align anisotropic Gaussians to surface curvature. EM would also introduce randomness, losing the



(a) overshoot (b) clumping

Figure 3.11: If the sampling rate is insufficient, or an overly large σ_h is used, this leads to overshoot (a): the Gaussians do not follow the distribution closely enough. In contrast, an overly small σ_h used for the seed Gaussians leads to a clumping artifact (b). Both problems can be avoided by using a sufficiently small step size h and setting $\sigma_h = h/\sqrt{8 \log 2}$.

guarantee that animated normal data will remain temporally coherent. Therefore we use a more direct approach specialized for our application.

Flat and curved elements. When creating the Gaussian mixture representing a surface heightfield, we can think about approximating the *heightfield* as locally first-order (flat), or we can approximate the *normal map* as locally first-order, essentially making the heightfield approximation second-order (curved). In the following, we will explore and compare both approaches. We will define *flat elements* as ones given by axis-aligned Gaussians (i.e. ones with a diagonal covariance matrix); intuitively, these represent flat surface patches with a constant normal. We also define *curved elements* to be given by general Gaussians; these represent patches with approximately first-order local normal variation, thus a second-order variation of the surface heightfield.

Seed points. As a first step, we distribute m seed points u_i in the normal map domain. Each of these will be converted into an element by considering the value, and optionally the derivative, of the normal map function at u_i. The simplest approach to choose the seed locations is uniform sampling with step h; the ideal value of h will depend on the frequency content of the normal map function.

Next, we need to choose the standard deviation σ_h of the Gaussians in the normal map domain. Consider two Gaussians with standard deviation of σ_h whose centers are h apart; we would like them to decay to half of their peak value exactly at the midpoint between them. Setting $\sigma_h = h/\sqrt{8 \log 2}$ achieves this, and works well in practice.

Flat elements. To convert a normal map into flat elements, we are approximating the

normal $n(u)$ as locally constant; that is, the surface is assumed locally *flat* near u_i. The i-th Gaussian we are creating can be written as:

$$G_i(u,s) = c_i \underbrace{\exp\left(-\frac{(u-u_i)^2}{2\sigma_h^2}\right)}_{\text{position band}} \underbrace{\exp\left(-\frac{(s-n(u_i))^2}{2\sigma_r^2}\right)}_{\text{normal band}}. \tag{3.17}$$

This results in a 2D Gaussian, whose inverse covariance matrix is diagonal, with values $1/\sigma_h^2$ and $1/\sigma_r^2$ on the diagonal. We can see the above 2D Gaussian as a product of a Gaussian band in positions around the sampling point u_i, and a Gaussian band in normals around the value $n(u_i)$. The resulting approximation can be seen in Figure 2.8(c): we can see how axis-aligned Gaussians are suboptimal for representing the position-normal distribution of a smooth surface.

Curved elements. For curved elements, we are intuitively trying to make the Gaussian locally align with the position-normal distribution, by approximating the normal map function $n(u)$ as linear (first-order) throughout the element. To achieve this, we replace the above normal band around $n(u_i)$ with a "sheared" band, which follows the first-order expansion $n(u) \approx n(u_i) + n'(u_i)(u-u_i)$. This leads to the following definition:

$$G_i(u,s) = c_i \exp\left(-\frac{(u-u_i)^2}{2\sigma_h^2}\right) \exp\left(-\frac{(s-n(u_i)-n'(u_i)(u-u_i))^2}{2\sigma_r^2}\right). \tag{3.18}$$

By using the shorthand notation $\delta u = u - u_i$, $\delta s = s - n(u_i)$ and $n' = n'(u_i)$, we can write this as:

$$G_i(u,s) = c_i \exp\left(-\frac{\delta u^2}{2\sigma_h^2}\right) \exp\left(-\frac{n'^2\delta u^2 - 2n'\delta u\delta s + \delta s^2}{2\sigma_r^2}\right). \tag{3.19}$$

This lets us write the 2×2 inverse covariance matrix of this Gaussian as follows:

$$\mathbf{\Sigma}_i^{-1} = \frac{1}{\sigma_h^2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{\sigma_r^2}\begin{pmatrix} n'^2 & -n' \\ -n' & 1 \end{pmatrix}. \tag{3.20}$$

The result can be seen in Figure 2.8(d): the position-normal distribution is approximated much better, with subtle artifacts still visible with 80 elements, but barely any error with 160 elements.

The seed size σ_h and the sampling step h need to be chosen well, otherwise approximation artifacts will result (Figure 3.11). The seed size can be fixed by following our σ_h heuristic, but the ideal step size will depend on the normal map frequency content. For well-sampled normal maps without excessive high frequencies, we found setting h to half a texel was sufficient for accurate results. More discussion of step size (and the resulting storage requirements) can be found in the next section.

Framework and mixture construction in 4D

Now that we have the flatland derivations, in this subsection, we transfer the flatland derivations to actual normal maps, and turn the framework into a full algorithm in 4D afterwards.

For a 2D normal map, we can similarly define the 4D position-normal distribution, and approximate it as a mixture of Gaussians:

$$\mathcal{N}(\mathbf{u},\mathbf{s}) = G_r(\mathbf{n}(\mathbf{u})-\mathbf{s}) \approx \sum_{i=1}^{m} G_i(\mathbf{u},\mathbf{s}), \tag{3.21}$$

where the G_i are 4-dimensional Gaussians in positions and normals. They can be represented as triples $(c_i, \mathbf{x}_i, \mathbf{\Sigma}_i)$ of a scaling coefficient, center and a 4×4 covariance matrix:

$$G_i(\mathbf{x}) = c_i \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{x}_i)\right), \tag{3.22}$$

where $\mathbf{x} = (\mathbf{u},\mathbf{s})^T$ is a 4D column vector. The constant c_i should be chosen such that the integral of G_i equals the area of the normal map it represents, e.g. 1/4 of a texel's area, when using 4 elements per texel. Again, we can evaluate the $\mathcal{P}$-NDFefficiently:

$$D_{\mathcal{P}}(\mathbf{s}) = \int_{\mathbb{R}^2} G_p(\mathbf{u})\mathcal{N}(\mathbf{u},\mathbf{s})\,\mathrm{d}\mathbf{u} \approx \sum_{i=1}^{m} \int_{\mathbb{R}^2} G_p(\mathbf{u}) G_i(\mathbf{u},\mathbf{s})\,\mathrm{d}\mathbf{u}, \tag{3.23}$$

where the latter integrand collapses to a 2D Gaussian, leading to a closed form solution. Please refer to the Appendix for details.

Like in flatland, we can create the Gaussian mixture by distributing seed points in texture space, and turning each seed point into either a flat or curved element, as detailed below.

Flat 4D elements. To convert a 2D normal map into flat elements, we define $\mathbf{s}_i = n(\mathbf{u}_i)$, $\delta\mathbf{u} = \mathbf{u}-\mathbf{u}_i$ and $\delta\mathbf{s} = \mathbf{s}-\mathbf{s}_i$. Extending the flatland idea to a 4D Gaussian is straightforward:

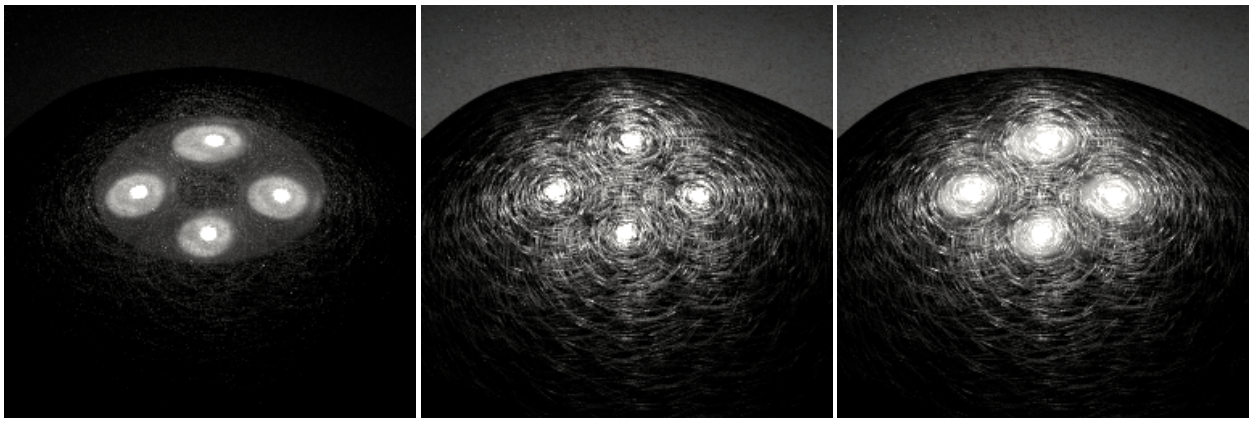
$$G_i(\mathbf{u},\mathbf{s}) = c_i \exp\left(-\frac{\|\delta\mathbf{u}\|^2}{2\sigma_h^2}\right)\exp\left(-\frac{\|\delta\mathbf{s}\|^2}{2\sigma_r^2}\right). \tag{3.24}$$

This covariance matrix can be easily seen to be diagonal: $\mathbf{\Sigma}_i^{-1} = \mathrm{diag}(\sigma_h^2,\sigma_h^2,\sigma_r^2,\sigma_r^2)^{-1}$.

As an aside, a mixture representing a collection of flat elements (as in metallic paint) does not necessarily need to be sampled from an underlying normal map. It can also be created directly: we can use Poisson sampling [27] to distribute seed Gaussians, and assign to each a random constant normal (e.g. drawn from the Beckmann distribution). This is used for our metallic paint flakes approximation.

Curved 4D elements. Here we are approximating the normal function $n(\mathbf{u})$ as linear: $n(\mathbf{u}) \approx \mathbf{s}_i + \mathbf{J}\delta\mathbf{u}$, where $\mathbf{J}$ is the Jacobian of n at $\mathbf{u}_i$. The Gaussian then becomes:

$$G_i(\mathbf{u},\mathbf{s}) = c_i \exp\left(-\frac{\|\delta\mathbf{u}\|^2}{2\sigma_h^2}\right)\exp\left(-\frac{\|\delta\mathbf{s}-J\delta\mathbf{u}\|^2}{2\sigma_r^2}\right). \tag{3.25}$$



(a) sampling (b) evaluation (c) combined

Figure 3.12: Our improved material model can be used inside a standard BRDF sampling/evaluation framework with multiple importance sampling. BRDF sampling alone (left) captures the reflection of the light through the flat areas of the map, but is suboptimal for rendering the scratches. Light sampling (middle), using our fast $\mathcal{P}$-NDFevaluation under the hood, captures illumination from the high-intensity parts of the HDR light texture onto the scratches. The combined result (c) has the benefits of both estimators.

By expanding $\|\delta\mathbf{s} - J\delta\mathbf{u}\|^2$, we find that the 4×4 inverse covariance matrix of this Gaussian can be written (using block notation) as:

$$\mathbf{\Sigma}_i^{-1} = \frac{1}{\sigma_h^2} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} + \frac{1}{\sigma_r^2} \begin{pmatrix} \mathbf{J}^T\mathbf{J} & -\mathbf{J}^T \\ -\mathbf{J} & \mathbf{I} \end{pmatrix}. \tag{3.26}$$

3.9 Implementation using Gaussian Elements

In this section, we will introduce key implementation using Gaussian elements. Specifically, we will discuss importance sampling and acceleration of the algorithm using a 4D hierarchy.

Importance sampling

The above sections dealt with $\mathcal{P}$-NDFevaluation, which is the more challenging part, but for integration in Monte Carlo rendering, we also need to be able to sample $\mathcal{P}$-NDFs on a given footprint $\mathcal{P}$.

We could sample the underlying normal map from which the Gaussian mixture is generated, and perturb the sampled normal by the intrinsic roughness. This is the same approach we use for generating a ground truth $\mathcal{P}$-NDF. An alternative way is to sample from the Gaussian mixture directly, picking an element proportional to its contribution to the footprint, then picking a normal from that element. The advantage of the first is speed and a simple implementation. However, it only works for mixtures that have been constructed from a

normal map, which need not be true in general (our metallic flakes in the top of Figure 3.16 are such an example). Furthermore, the latter method samples a distribution that matches our evaluation exactly, instead of just approximately. Therefore, we use the latter option in our results.

Having the ability to evaluate and sample the $\mathcal{P}$-NDFenables us to fully integrate the algorithm into a multiple importance-sampling framework [129], and support various sources of illumination: environment maps and area lights, including ones with HDR emission textures. Figure 3.12 shows a scratched surface rendered under a textured area light. We separate the two estimators (sampling and evaluation); the components include the weights, so that the sum of the separated images is the combined image. The separation we observe supports natural intuition: the reflection of the light through the flat parts of the scratched map is easy to render by BRDF sampling (a), but shading the scratches requires sampling the high-intensity spots on the lightsource and evaluating the $\mathcal{P}$-BRDF and thus the $\mathcal{P}$-NDFin that direction (b). The combined image (c) has the benefits of both.

Acceleration hierarchy

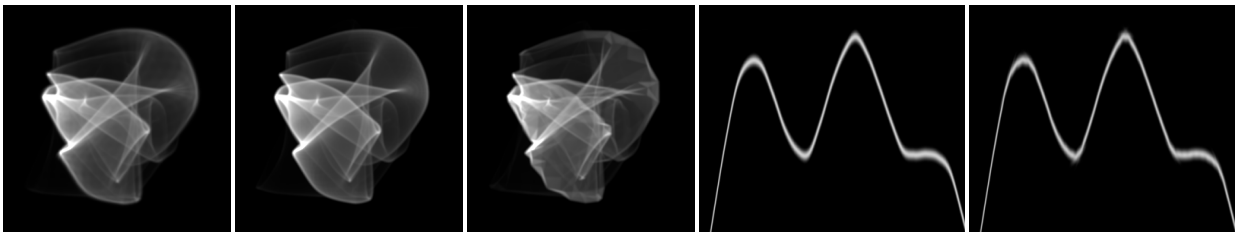
The number m of 4D Gaussian elements in our mixture will be in the millions: a typical sampling step h is about half texel to one texel, which for a 2048^2 normal map will produce 16 million elements. Clearly, only a tiny number of these Gaussians will have a non-negligible contribution to a given query footprint $G_p(\mathbf{u})$ and a given query half-vector $\mathbf{s}$. This problem is identical to one faced by Jakob et al. [55]and our triangulation-based approach earlier, except the primitives they are bounding are point reflectors and triangles, respectively. They address this issue by 4D bounding-volume hierarchies over the (u, v, s, t)-space, and we follow a similar approach with some improvements.

A straightforward approach would be to create a 4D bounding box for each Gaussian element (treating their contribution as negligible beyond some distance, e.g. using a 3σ rule), and build a hierarchy on these bounding boxes in a top-down manner, using median splits. A $\mathcal{P}$-NDFevaluation query in this hierarchy is given by a rectangle in (u, v)-space, bounding the footprint Gaussian $G_p(\mathbf{u})$, and a point in (s, t) space, specifying the half-vector of interest. The contributing Gaussians can be found by a top-down traversal, pruning bounding boxes with no intersection with the query.

We found this natural approach works, but its performance can be enhanced by two additional ideas. First, we can trade off storage for performance: instead of having a single hierarchy, we subdivide (s, t)-space into a few hundred cells (sub-domains), and build a much smaller hierarchy per cell. Our query is point-wise in (s, t), so each query can be immediately answered by a single one of these smaller sub-hierarchies, and can therefore be faster. Of course, this is at the expense of extra storage, since an element will commonly occur in multiple sub-hierarchies.

An additional speedup can be achieved by stopping the build process earlier (at about 5 Gaussians per leaf node), since our closed-form solution per Gaussian is fast, and at some level becomes less of a bottleneck than traversal operations.

Finally, since our $\mathcal{P}$-NDFsampling is done using the element approximation, instead of sampling the underlying normal map, we need to accelerate the sampling operation. Here we just need to sample the footprint Gaussian, obtaining a (u, v) pair, and then quickly find all elements that contribute to this pair. We simply use a separate 2D hierarchy over the (u, v) domain to answer this query.



(a) 24 sec Gaussian elem. (b) 3114 sec Triangulation 32 Δ/texel (c) 300 sec Triangulation 2 Δ/texel (d) Ground truth Slice of $\mathcal{N}(\mathbf{u}, \mathbf{s})$ (full 4D) (e) Our approx. Slice of $\mathcal{N}(\mathbf{u}, \mathbf{s})$ (full 4D)

Figure 3.13: ***Left 3 images:*** *$\mathcal{P}$-NDFimages evaluated in single-threaded C++; the timings are for evaluating the visualizations themselves, not final renderings. Our method (a) using Gaussian mixtures is about 130× faster than our triangulation-based solution when the normal map is discretized using 32 triangles per texel (b). The match is very good, because our Gaussian mixture fits the position-normal distribution well. Our method is, moreover, still more than 12× faster than our previous approach if using only 2 triangles per texel (c), and has higher quality.* ***Right 2 images:*** *Accuracy of fitting the actual 4D position-normal distribution $\mathcal{N}(\mathbf{u}, \mathbf{s})$ with our Gaussian mixture. The full distributions are four-dimensional and cannot be visualized directly. Therefore we slice the space by taking a small segment of a normal map scanline, and show only the s-component of normals, integrating along the t-component. Note how our approximation is accurate, with minimal overshoot.*

3.10 Results using Gaussian Elements

We implemented the improved algorithm in C++. For rendering final images, we integrated our $\mathcal{P}$-NDFevaluation and sampling code in the Mitsuba framework [51].

Correctness and performance

First, we compare the $\mathcal{P}$-NDFs computed using our method to our earlier solution, which is accurate and can be treated as ground truth. Figure 3.13 (left images) shows a good match using our improved $\mathcal{P}$-NDFevaluation, while demonstrating a 130× speedup. One may think that the speed-up is because of overly fine subdivision (32 triangles per texel). However, it turns out that decreasing the subdivision rate of our previous method significantly reduces the quality and is still 12× slower than our method.

triangulation Gaussian elements triangulation Gaussian elements

Figure 3.14: Rendering comparison to our previous triangulation-based solution (with full quality settings), demonstrating that a close match in the $\mathcal{P}$-NDFvalues translates to a close match in renderings using our improved Gaussian elements approach. We separate the specular component of the image and use an identical sampling pattern for a direct comparison. Top: direct specular component on a simple curved surface, showing a match down to specific glints. Bottom: cutlery scene with additional light transport added. We achieve a speedup of about 32× in rendering the specular component.

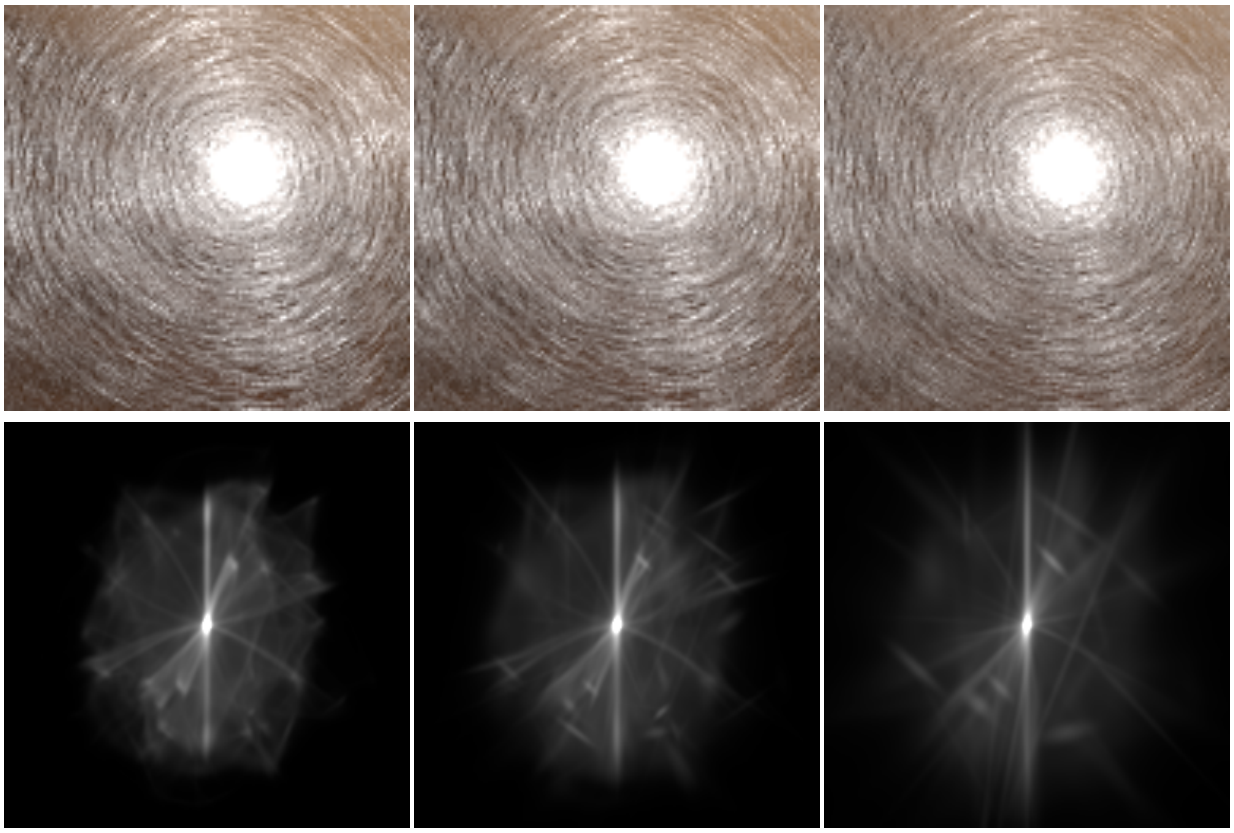
In addition to comparing $\mathcal{P}$-NDFs, we also directly compare the ground truth position-normal distribution $\mathcal{N}(\mathbf{u}, \mathbf{s})$ to our approximation using Gaussian mixtures. This can be seen in Figure 3.13 (right images). Since these are 4D distributions, we visualize them by slicing along a small segment of a normal map row, and integrating along the t-component of the normal. This shows that our approximation matches the ground truth well, if the right sampling step is chosen.

We also compare to our basic triangulation-based approach on an actual rendering in Figure 3.14. Here, for a direct comparison, we separate the specular component and use a point light like our previous method; we also use the same sampling pattern. We get closely matching results, with a speedup of 32× in computing the direct specular component (this is less than in Figure 3.13, because other operations become important, notably tracing eye and shadow rays).

Effect of sampling rate on quality

Of course, a key issue is whether the step size h delivers sufficient accuracy, and this depends on the specific normal map, and on the desired intrinsic roughness σ_r. In our experience, setting h to half a texel size delivers accurate results for $\sigma_r = 0.005$, for typical normal maps that do not contain excessive fine noise. We use these settings in our final image and video results.

Figure 3.15 shows the result of varying the step size h, setting it to 0.5, 1, and 2 texels. This corresponds to 4, 1, and 0.25 elements per texel, respectively. With 4 elements per texel, both the image and the $\mathcal{P}$-NDFvisualization show no artifacts. With one element per texel, some error can be seen in the $\mathcal{P}$-NDFvisualization but the rendering itself is barely



(a) $h = 0.5$ texel (b) $h = 1$ texel (c) $h = 2$ texels

Figure 3.15: Comparison of different sampling steps h. When setting h to half a texel (a), i.e. 4 elements per texel, neither the image nor the $\mathcal{P}$-NDFvisualization show approximation artifacts; this setting is what we use in our final results. When $h = 1$ texel (b), i.e. one element per texel, the rendering is only slightly degraded but some individual Gaussians become visible in the $\mathcal{P}$-NDFvisualization (b). With $h = 2$ texels (c), i.e. one element per 4 texels, the rendered highlight starts showing signs of lower contrast, while the $\mathcal{P}$-NDFimage clearly shows overshoot and individual Gaussians.

degraded. With one element per 4 texels, the $\mathcal{P}$-NDFimage shows obvious overshoot and individual Gaussians; this causes directional blurring, whose effect on the rendering is some loss of specular contrast.

Effect of element sampling rate on quality

Of course, a key issue is whether the step size h delivers sufficient accuracy, and this depends on the specific normal map, and on the desired intrinsic roughness σ_r. In our experience, setting h to half a texel size delivers accurate results for $\sigma_r = 0.005$, for typical normal maps that do not contain excessive fine noise. We use these settings in our final image and video

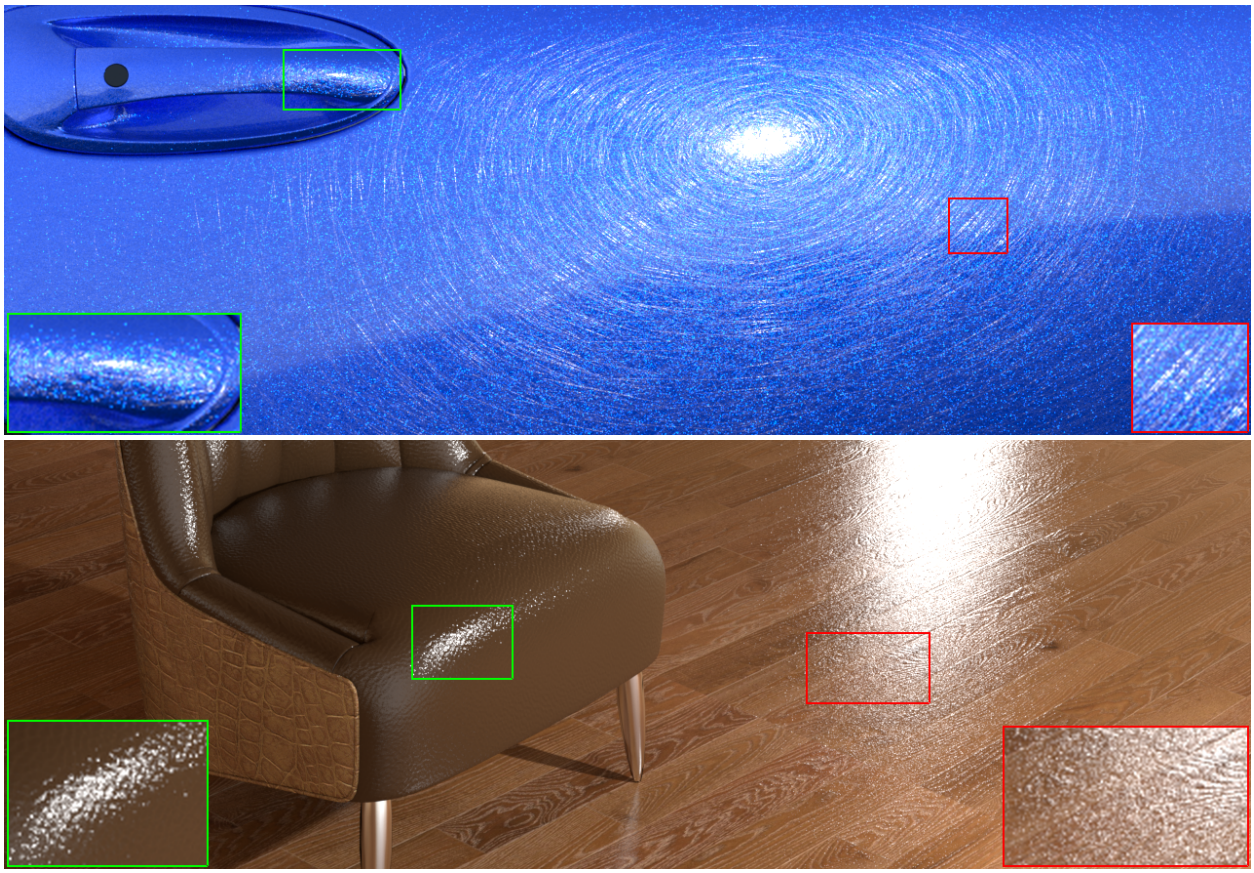


Figure 3.16: Final image renderings using our improved method. Insets are zooms of image itself (not re-rendered at high resolution). ***Top:*** *car paint (scratched coating with embedded metallic flakes). Both effects use our method; the scratches use curved elements and the metallic paint uses flat elements.* ***Bottom:*** *leather sofa on a wooden floor; both materials use a combination of a macro-level standard normal map, and a micro-level map handled using our technique.*

results.

Figure 3.15 shows the result of varying the step size h, setting it to 0.5, 1, and 2 texels. This corresponds to 4, 1, and 0.25 elements per texel, respectively. With 4 elements per texel, both the image and the $\mathcal{P}$-NDFvisualization are accurate. With one element per texel, some error can be seen in the $\mathcal{P}$-NDFvisualization but the rendering itself is barely degraded. With one element per 4 texels, the $\mathcal{P}$-NDFimage shows obvious overshoot and individual Gaussians; this causes directional blurring, whose effect on the rendering is some loss of specular contrast.

Final renderings

We also illustrate our method's capabilities on final image renderings, shown in Figure 1 and 3.16. Results of this kind would not be easily achievable by the method of Jakob et al. [55](due to lack of support for smooth explicit normal maps), or our triangulation-based method (due to lack of support for high-frequency environment map and area lighting). Please make sure to watch the temporal versions of these results in the included video.

Kettle. This scene (Figure 1) demonstrates a glinty scratched stainless steel material similar to one shown previously for the basic approach. However, note that our rendering is lit by area and environment lighting without multiple passes or other special handling. In fact, our performance is only 1.4× slower than the same scene rendered with a traditional microfacet BRDF with no glinty behavior (2.65 vs. 1.88 min).

Car door. This scene is shown in Figure 3.16, top. The material combines two different glinty effects. A top coating is modeled by a scratched normal map, represented using curved elements. The bottom layer of metallic flakes embedded in the paint under it is represented using flat elements and has been created directly by Poisson sampling, without the need for a normal map. Lighting is from an environment map combined with a point light.

Wood floor and leather sofa. This scene is shown in Figure 3.16, bottom. It shows a leather sofa on a wood floor. It is demonstrating an additional useful feature of combining a macro-scale bump map (handled using the standard approach of shading frame perturbation) with a micro-scale normal map handled using our $\mathcal{P}$-BRDF approach, similar to [155]. This lets our method render materials that have interesting structure at multiple scales, which would require immensely large normal maps to represent using a naive single-map approach.

Time and storage. All images are rendered at 1280×720, with 1024 samples per pixel (1600 for leather sofa). These sampling rates were used to achieve low Monte Carlo noise in the videos; good still images can be produced with fewer samples. The timings on a 36-thread Amazon EC2 machine (c4.8xlarge) were 2.6 min (kettle), 6.8 min (car door) and 7.6 min (leather sofa).

The storage requirements depend on normal map size and sampling rate. Our results use a 2048^2 texture and 4 elements per texel, which requires 720M to store the Gaussians and an additional 400M for the acceleration hierarchy. If using 1 element per texel (which may well be sufficient for most applications), the costs would be 180M and 100M respectively. The normal map itself would take 32M (in floating point precision).

3.11 An Extension using Wave Optics

So far, we have presented a triangulation-based approach and an improved Gaussian elements approach for accurate and efficient rendering of detailed surfaces. Theoretically, with either of these methods, we should be able to match real world appearance exactly. However, when we look at the real photos in Figure 3.18, we notice the colors even if they are illuminated by a white light source. This is an interesting phenomenon that we immediately conclude as

geometric optics | wave optics single wavelength | wave optics spectral

Figure 3.17: The Cutlery *scene rendered using wave optics. Left: Rendering with our improved geometric method. Middle: Using wave optics, even with a single fixed wavelength, our method generates a more natural appearance as compared to geometric optics. Right: A spectral rendering additionally shows subtle but important color glint effects. Insets show enlarged regions and representative BRDFs generated using each method. We encourage readers to zoom in to better see color and detail, and to view the full resolution supplementary images to see the subtle details on all of the figures.*

impossible for traditional geometric optics—it will produce exactly white highlights under white light source.

This observation leads to three questions. First, if geometric optics has its limitations at such fine level of detail, is wave optics able to give us more accurate results? Second, can we design a rendering algorithm based on the more accurate wave optics models, but also able to compute spatially-varying solutions with high-resolution detail? Third, how close is geometric optics to wave optics, apart from the color difference? To our knowledge, our work is the first to consider all these questions in full generality, modeling the surfaces as arbitrary discretized heightfields.

In this section, we discuss how to evaluate the detailed BRDF integrals for our wave optics diffraction models. Our high-level idea for efficiently approximating the integral in equation (2.8) is to approximate the phase-delay reflection function $R^\star(\boldsymbol{s})$ by a weighted combination of *Gabor kernels*, which are products of a 2D Gaussian with a complex exponential (plane wave). These kernels are well suited to representing the high-frequency features found in typical $R^\star(\boldsymbol{s})$, while also having other desirable properties.

Before we proceed, we briefly list the symbols used in our wave optics models in Figure 3.19. Detailed explanations of these symbols, together with the background knowledge for wave optics, are introduced in Chapter 2.

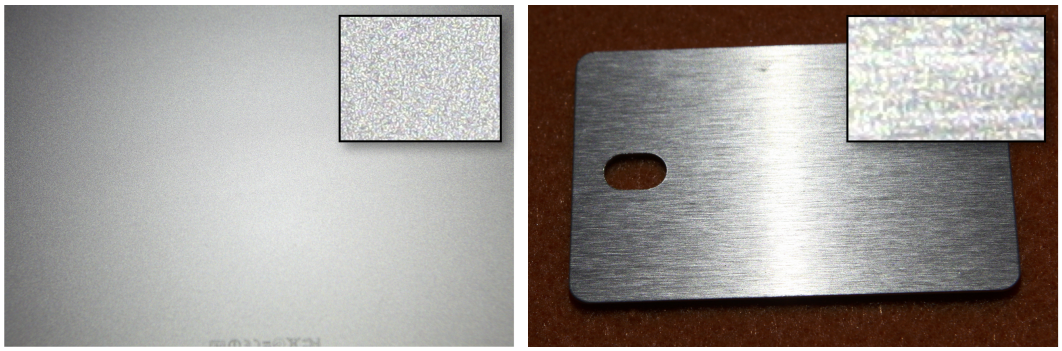


Figure 3.18: Real world photos showing colors under a white light. Left: the back of a laptop. Right: an aluminum patch. Both photos are taken with a Canon EOS 5D Mark III ESLR with a small LED light. Note how these materials produce colors even if they are generally believed to be white.

i	Imaginary unit for complex numbers, $i^2 = -1$
λ	Wavelength of light
$\boldsymbol{n}$	Average surface normal (equal to z-axis)
$\boldsymbol{s}$	2D point (on the XY plane)
$H(\boldsymbol{s})$	Height of surface above $\boldsymbol{s}$
$\boldsymbol{H}'(\boldsymbol{s})$	Gradient of height function
$\bar{\mathcal{S}}$	Domain of height function (region on XY plane)
$A_{\bar{\mathcal{S}}}$	Area of $\bar{\mathcal{S}}$
$\boldsymbol{\omega}_\mathrm{i}$	Direction from which light arrives (3D unit vector)
$\boldsymbol{\omega}_\mathrm{o}$	Direction of reflected light (3D unit vector)
$\boldsymbol{\psi}$	$\boldsymbol{\psi} = \boldsymbol{\omega}_\mathrm{i} + \boldsymbol{\omega}_\mathrm{o}$
$\overline{\boldsymbol{\psi}}$	2D projection $\boldsymbol{\psi}$ (removing its z-component)
f_r	Bidirectional reflectance distrib. function (BRDF)
F	Surface reflectance (e.g., from Fresnel equations)
ξ_1, ξ_2, ξ_3	See figure 2.5

Figure 3.19: List of symbols used in our wave optics models.

Gabor kernels

Let us define a Gabor kernel as the product of a 2D Gaussian and a complex exponential:

$$g(\mathbf{s}; \boldsymbol{\mu}, \sigma, \boldsymbol{a}) = G_{2D}(\mathbf{s}; \boldsymbol{\mu}, \sigma)\, e^{-i2\pi(\boldsymbol{a}\cdot\mathbf{s})} \tag{3.27}$$

where $G_{2D}(\mathbf{s}; \boldsymbol{\mu}, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|\mathbf{s}-\boldsymbol{\mu}\|^2}{2\sigma^2}\right)$ is a normalized 2D isotropic Gaussian. Here $\boldsymbol{\mu}$ is the center, σ the width and $\boldsymbol{a}$ the plane wave parameter. This definition is similar to others used in the literature; the normalization constant of the Gaussian and the additional 2π factor in the complex exponential are chosen to simplify the following derivations.

The Fourier transform of a Gabor kernel can be written as another Gabor kernel:

$$\begin{aligned}\mathcal{F}[g(\mathbf{s};\boldsymbol{\mu},\sigma,\boldsymbol{a})](\boldsymbol{v}) &= e^{-i2\pi\left(\boldsymbol{\mu}\cdot(\boldsymbol{v}+\boldsymbol{a})\right)}\, e^{-2\pi^2\sigma^2\|\boldsymbol{v}+\boldsymbol{a}\|^2} \\ &= \frac{1}{2\pi\sigma^2} e^{-i2\pi(\boldsymbol{\mu}\cdot\boldsymbol{a})}\, g\left(\boldsymbol{v};-\boldsymbol{a},\frac{1}{2\pi\sigma},\boldsymbol{\mu}\right) \end{aligned} \tag{3.28}$$

Notably, Gabor kernels have an analytical Fourier transform that is itself a Gabor kernel. This means that the kernels and their transforms both have spatially localized support (ignoring negligibly small values of the Gaussian component), which is a key property for designing an efficient pruning algorithm.

Approximating Phase Shifts with Gabor kernels

As explained in Chapter 2, the highfield introduces spatially-varying phase shifts $R(\mathbf{s})$ for different wavelengths (Equation 2.6). However, the function R is highly complicated for further computation. So, we approximate R with Gabor kernels. We first subdivide the heightfield domain $\bar{\mathcal{S}}$ into a grid of cells. We use a uniform grid so all the cells are identically-sized squares, matching the original heightfield texels, but an adaptive subdivision could also be used. Then we select a set of cells, with centers $\boldsymbol{m}_k$, that covers the support of the current coherence kernel. Since the cells are much smaller than the coherence area, we approximate the coherence kernel as being constant over a cell with value $w_k = w(\boldsymbol{m}_k - \boldsymbol{x}_c)$. Then we place a Gabor kernel centered on each grid cell designed to approximate $R(\boldsymbol{s})$ in its neighborhood. Together this gives us an approximation for $R^\star(\boldsymbol{s})$ of the form:

$$R^\star(\boldsymbol{s}) \approx \sum_k w_k R_k(\mathbf{s}) = \sum_k w_k C_k\, g(\mathbf{s};\boldsymbol{m}_k,\sigma_k,\boldsymbol{a}_k) \tag{3.29}$$

where C_k is a complex constant, incorporating an appropriate scaling coefficient and phase shift.

We choose $\sigma_k = l_k/2$, where l_k is the side length of the cell. This choice was found to give good results experimentally. A sum of Gaussians is not an exact partition of unity; this leads to slight approximation error that manifests itself as spurious periodic copies of the main transform image in the Fourier domain. However, for the choice $\sigma_k = l_k/2$, the copies are weak enough that we do not observe them in practice.

Next, we approximate the heightfield $H(\mathbf{s})$ in each cell by its first order expansion around $\boldsymbol{m}_k$:

$$H(\mathbf{s}) \approx H(\boldsymbol{m}_k) + \boldsymbol{H'}(\boldsymbol{m}_k)\cdot(\mathbf{s}-\boldsymbol{m}_k) \tag{3.30}$$

$$= \boldsymbol{H'}(\boldsymbol{m}_k)\cdot\mathbf{s} + (H(\boldsymbol{m}_k) - \boldsymbol{H'}(\boldsymbol{m}_k)\cdot\boldsymbol{m}_k) \tag{3.31}$$

where $\boldsymbol{H'}(\boldsymbol{m}_k)$ is the gradient of the heightfield at $\boldsymbol{m}_k$. Substituting this approximation into the definition of $R(\boldsymbol{s})$, we can approximate a single grid cell's contribution as:

$$R_k(\mathbf{s}) = B_{2D}(\mathbf{s};\boldsymbol{m}_k,l_k)\,\xi_2\, e^{-\frac{i2\pi\xi_3}{\lambda}H(\mathbf{s})} \tag{3.32}$$

$$\approx l_k^2 G_{2D}(\mathbf{s};\boldsymbol{\mu}_k,\sigma_k)\,\xi_2\, e^{-\frac{i2\pi\xi_3}{\lambda}\left(\alpha_k+\boldsymbol{H'}(\boldsymbol{m}_k)\cdot\mathbf{s}\right)} \tag{3.33}$$

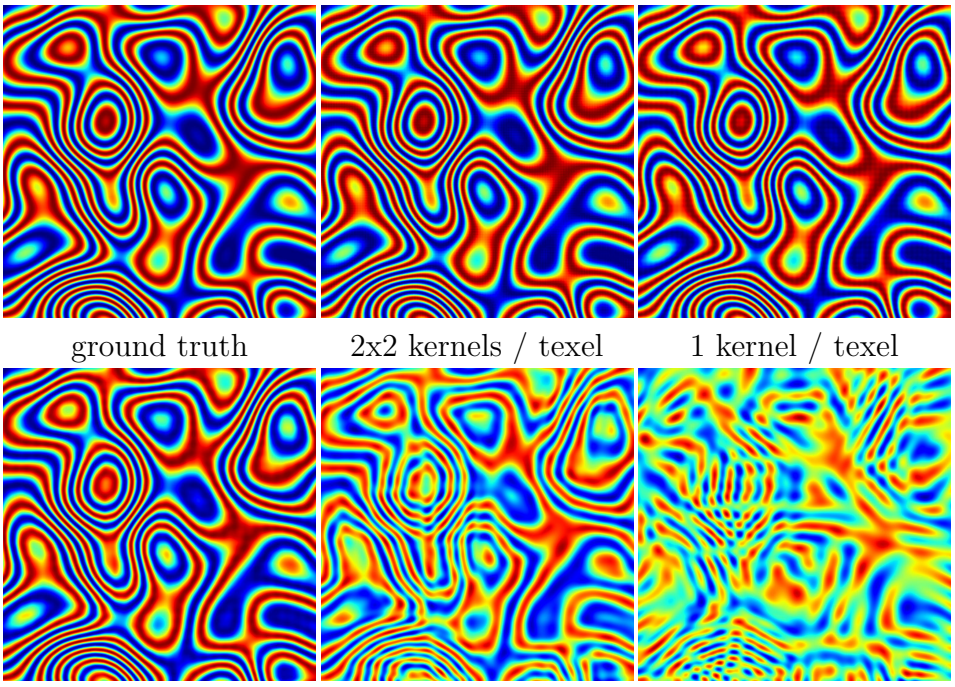


Figure 3.20: A color-mapped (range [-1,1]) visualization of the real component of $R(\mathbf{s})$ for the isotropic noise heightfield (the imaginary component looks similar). The area depicted is about 64×64 texels, using the resolution of 1 micron / texel. Note the common structure seen in these functions: high-frequency ripples aligned with slopes of the original heightfield, with frequency increasing proportional to slope. This structure is ideal for approximation by Gabor kernels. These images show the approximation quality for various kernel sampling densities. All our results use 1 kernel per texel (i.e. per micron). Note that as the number of kernels decreases, the approximation degrades, as expected.

where $\alpha_k = H(\boldsymbol{m}_k) - \boldsymbol{H'}(\boldsymbol{m}_k) \cdot \boldsymbol{m}_k$. B_{2D} is a binary box function indicating the domain of the grid cell, which integrates to the cell's area l_k^2. Then we replace the box function with a 2D Gaussian of the equal area.

Comparing the coefficients in Equation 3.33 with the definition in Equation 3.29, we have

$$C_k = l_k^2 \xi_2 e^{-\frac{i2\pi\xi_3}{\lambda}\left(H(\boldsymbol{m}_k) - \boldsymbol{H'}(\boldsymbol{m}_k)\cdot\boldsymbol{m}_k\right)} \tag{3.34}$$

$$\boldsymbol{a}_k = \frac{\xi_3 \boldsymbol{H'}(\boldsymbol{m}_k)}{\lambda} \tag{3.35}$$

which completes our Gabor approximation for $R^\star(\boldsymbol{s})$.

Figure 3.20 shows $R(\mathbf{s})$ for an example heightfield compared to its approximation as a sum of Gabor kernels. While the sampling rate is slightly visible in the approximation, overall it closely matches the high frequency details in the original.

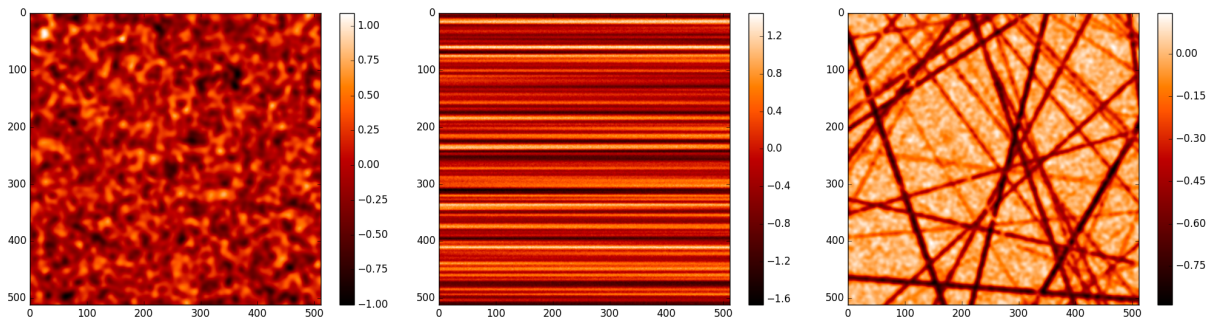


Figure 3.21: The heightfields used in our wave optics model. Left to right: isotropic bumps, brushed metal, scratched metal. These are 512^2 crops of the full 8192^2 maps. The units (horizontal and vertical) are microns (μm), so the full maps cover a square area about 8.2 mm × 8.2 mm large.

BRDF approximation

Finally we use our Gabor kernel approximation to evaluate the BRDF. Starting from Equation 2.11 we have:

$$f_{\mathrm{r}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\xi_1}{A_{\mathrm{c}}} \left| \widetilde{R^\star} \left(\frac{\overline{\boldsymbol{\psi}}}{\lambda} \right) \right|^2 \tag{3.36}$$

$$\approx \frac{\xi_1}{A_{\mathrm{c}}} \left| \sum_k w_k C_k \mathcal{F}\left[g(\mathbf{s}; \boldsymbol{m}_k, \sigma_k, \boldsymbol{a}_k) \right] \left(\frac{\overline{\boldsymbol{\psi}}}{\lambda} \right) \right|^2 \tag{3.37}$$

where we can use the above definitions of the quantities C_k and $\boldsymbol{a}_k$, and equation (3.28) to evaluate the Fourier transform of the Gabor kernel. Thus we can evaluate the sum in a straightforward manner by iterating over all the cells within the coherence area. We also apply pruning to non-contributing cells, as detailed in the next section.

3.12 Results using Wave Optics

Implementation Details

Before we show our results using wave optics, we first provide key implementation details of our Gabor kernel solution.

Heightfields and Gabor kernels. We use pre-defined high resolution ($8K \times 8K$) heightfields as texture maps to specify the microgeometry, where each texel represent a fixed size of 1 square micron in the real world. The heightfields are tiled repeatedly to achieve a high resolution over a surface. The texels in a heightfield form a uniform grid naturally, so, we convert each texel into a Gabor Kernel, as specified in Section 3.11.

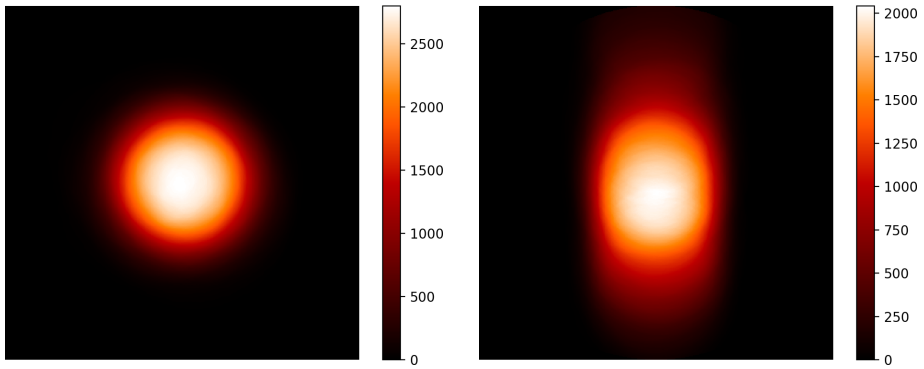


Figure 3.22: Visualization of the numbers of Gabor kernels that are evaluated to calculate the BRDF values toward different directions. Note that the shapes of the corresponding BRDFs are captured well, and that a large number of evaluations are successfully pruned.

For simplicity, we assume no distortion from the texture map, i.e. the texture coordinates are defined to be area preserving and orthogonal in terms of u and v directions in the world coordinate system.

Acceleration by pruning. To accelerate computation, the key is to quickly decide whether a Gabor kernel contributes to the desired outgoing direction $\boldsymbol{\omega}_{\mathrm{o}}$. Regardless of the cancellation from the complex numbers, each Gabor kernel is bounded by a Gaussian $G_{2D}(\mathbf{s}; \boldsymbol{m}_k, \sigma_k)$ positionally and by $G_{2D}(\boldsymbol{v}; -\boldsymbol{a}_k, \frac{1}{2\pi\sigma_k})$ directionally. Although in theory Gaussians have infinite support, in practice we limit them to within ± 3 standard deviations, and clamp them to zero outside this region so they have only localized support.

We pre-generate a mipmap-style hierarchy for each heightfield, where each node contains both positional and directional bounding boxes of its 4 child nodes. For each BRDF query, we perform a top-down traversal of this hierarchy, discarding nodes that are not within the coherence region $\bar{\mathcal{S}}$ using their positional bounding boxes. At the same time, we use the directional bounding boxes to prune the nodes that will not contribute to the query direction $\overline{\psi}/\lambda$.

Figure 3.22 shows the number of evaluations towards different directions to generate an example BRDF image. In general, each Gabor kernel contributes to a much larger range directionally in wave optics than the elements in geometric optics [146]. This explains the soft appearance in these images, as well as slower performance of wave optics. However, our hierarchical pruning is still efficient. In practice, we have a more than $50\times$ speedup as compared to the un-accelerated implementation.

Importance sampling. With the Gabor kernels defined to represent a heightfield, it is straightforward to perform BRDF importance sampling to get the outgoing ray for global illumination. First, we randomly pick a Gabor kernel within the coherence region according to its weighting function $w(\mathbf{s})$. Then, we immediately know that the chosen Gabor kernel contributes to a Gaussian directionally, as analyzed in the acceleration part. By sampling

this Gaussian, we have the sampled query direction $\overline{\psi}/\lambda$ and thus the corresponding outgoing direction $\boldsymbol{\omega}_\mathrm{o}$.

The sampling weight can be calculated as the BRDF evaluation with sampled $\boldsymbol{\omega}_\mathrm{o}$, divided by the sampling pdf. However, in practice, we found that wave optics effects are essentially not observable in indirect lighting. So, we assume that our sampling weight is always 1, i.e. discarding the complex cancellations and assuming contribution only from the Gaussian part of each Gabor kernel. This gives us significant speed-up, allowing the indirect illumination to use more samples to converge. This is roughly equivalent to reducing the coherence area used for indirect illumination to the size of our Gabor kernels.

Practical rendering pipeline. For convenience, we separate the final rendered image into three components. First, direct illumination from point lights. Second, indirect illumination from point lights. Third, illumination from other lights, including the environment lighting, both direct and indirect. The separation allows us to use very few samples per pixel to render the most time-consuming first part, usually only 4 to 25 samples on a regular sub-pixel grid.

Spectral rendering. For each BRDF evaluation, we compute for different wavelengths ranging from 0.36 microns to 0.83 microns, i.e. the visible spectrum. We find that using 8 spectral samples is generally good enough to produce identical results to those generated using more samples. We split the wavelength range into bins and use the midpoints (not endpoints) of the bins as the samples. We follow the standard spectrum samples→XYZ→RGB method to eventually convert the spectral values to the sRGB color space.

We also find it useful to perform the top-down pruning only once using the largest wavelength. In this way, we record all contributing Gabor kernels first, then evaluate them for all spectrum samples at once. As a result, our computation time scales sub-linearly with the number of spectrum samples, which gives us another 3× speedup, compared with brute force spectral rendering.

Heightfield and BRDF visualizations

Figure 3.21 shows a colormap visualization of the heightfields used in our results. For all heightfields, we use a discretization step of 1 micron. The heightfields were generated procedurally by inverse FFT noise generation, and (in the case of scratches) by drawing lines with randomized positions, depths, and widths.

In Figure 3.23, we show visualizations of the outgoing BRDF lobes of our model and geometric optics, for a fixed incoming direction and footprint (coherence area). This illustrates the differences between geometric and wave optics, and also the differences between a single-wavelength and spectral simulation. Note that the appearance of high-frequency features is clearly different in the geometric and wave solutions: the geometric solutions contain sharp folds in areas where the normal map Jacobian becomes singular [147]. The wave optics solutions have no such features, and the high frequencies in them are more reminiscent of laser speckle. Also note the significant color effects in the full spectral wave optics version.

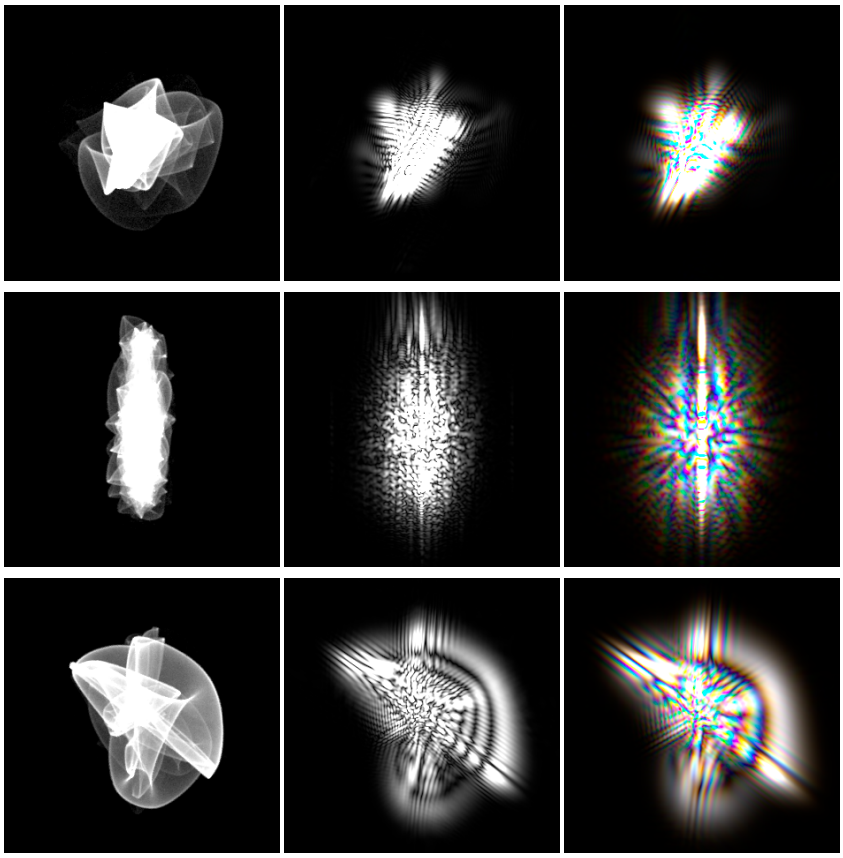


Figure 3.23: Visualizations of the outgoing BRDF lobes on the projected hemisphere. Top: isotropic bumps, middle: brushed, bottom: scratched. Note the clearly different high-frequency features predicted by geometric and wave optics. Also note the significant color effects predicted by wave optics (here we are using 8 spectral samples).

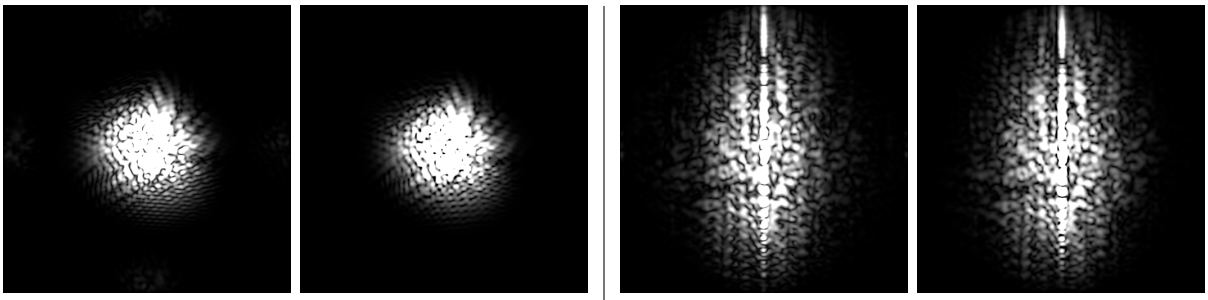
Ground truth comparison

In Figure 3.24, we show BRDF lobes computed with our approach (for a single wavelength) side-by-side with lobes computed using the FFT algorithm applied to equation 2.11. Note the close match, despite our method taking a completely different approach of Gabor kernel approximation.

Rendered results

In this subsection, we illustrate our method's capability to render actual scenes using wave optics, as shown in Figures 3.17, 3.26, 3.25, and 3.27.

Scene configurations and performance comparisons are listed in Table 3.3. In general, for



isotropic bumps | brushed metal

Figure 3.24: Comparison of BRDF lobes computed using our Gabor kernel approach (left image in each pair) to ground truth computed by evaluating equation 2.11 using the FFT algorithm (right image).

Scene	Patch	Cutlery	Laptop	Tumbler
# Point light(s)	1	1	1	2
# Env. light	0	1	1	1
Material	Al	Ag	Al	Fe
# Samples (direct)	4	9	25	25
# Samples (ind.+env.)	N/A	256	1024	1024
Direct (geom.)	9.6s	3.1s	37.4s	19.8s
Direct (single)	3.7m	0.8m	6.4m	1.9m
Direct (spectral)	13.1m	2.4m	21.1m	6.4m
Indirect + env.	N/A	4.0m	25.1m	9.7m
All (geom.)	N/A	4.2m	25.7m	10.0m
All (single)	N/A	4.8m	31.5m	11.6m
All (spectral)	N/A	6.4m	46.2m	16.1m

Table 3.3: Scene configurations including materials of the main objects and number of samples per pixel, and performance comparisons between geometric optics and wave optics with 1 *and* 8 *spectral samples. For the performance of the* Patch *scene, we use the isotropic noise heightfield as representative.*

direct illumination from point lights, our method with a single wavelength is about $5-20\times$ slower than geometric optics, and about another $3.5\times$ slower with 8 spectral samples. This is because of the wide directional spread predicted by wave optics, as analyzed earlier. However, direct illumination only takes up about half of the overall computing time. Considering indirect lighting and environment lighting together, the performance of our wave optics method is within $1.5\times$ of geometric optics, and is thus a practical solution. In the rest of this subsection, we will discuss individual scenes.

Patch. This is a simple scene showing a 5 $cm \times 5$ cm patch. The camera is looking towards the center of the patch from an elevation angle of 45°. The point light is on the

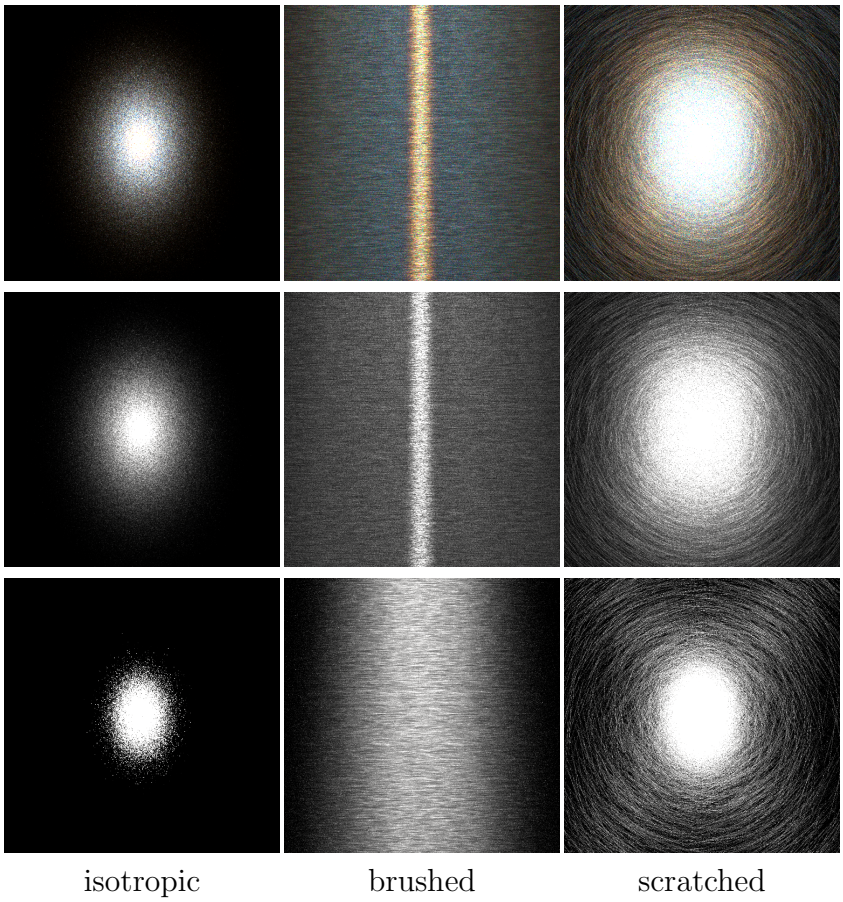


Figure 3.25: The Patch *scene showing renderings of different heightfields with a point light. (Top row) Spectral. (Middle row) Single wavelength. (Bottom row) Geometric.*

opposite side, and moves left and right in the video. Figure 3.25 shows renderings of three different heightfields (isotropic noise, brushed and scratched), each rendered using multiple wavelengths, single wavelength (0.4 microns) and geometric optics for comparison. We added isotropic noise on top of the brushes and scratches to make them more visible under the point light.

From these images, we can clearly see that our method is able to produce characteristic structures from the underlying heightfields: intuitively, round highlight for isotropic, vertical anisotropic highlight for brushed and spiderweb-like highlight for scratched. These shapes indicate the correctness of our method. Also, since different wavelengths behave differently in wave optics, colors are expected from spectral rendering.

Cutlery. This scene shows silver cutlery with strong scratches, rendered using a point light with static greyscale environment lighting, in order to make sure that the colors are from diffraction. In Figure 3.17, we can clearly see the colored scratches rendered using multiple wavelengths. Also, even with a single wavelength, our method is able to generate a

Figure 3.26: Left: Rendering of a laptop with a point light and environment lighting using our method. Top right: Close up rendering of the corner of the laptop with the same lighting condition. Bottom right: A photograph of a MacBook (around 20 cm × 4 cm region) lit by a small LED light in a dark room. Our method is able to produce appearance that is perceptually similar to the photograph, showing colored glints from the underlying noisy microstructure of the aluminum laptop body.

more convincing result, as we compare with the geometric method by Yan et al. [146]. The geometric method arguably produces harsher glints, due to the sharper folds in the BRDF lobes predicted by the $\mathcal{P}$-NDF theory.

Laptop. This scene shows a laptop with a roughened aluminum matte finish (modeled as a Gaussian random heightfield). It is rendered using a point light and environment lighting. We can observe colored glints in Figure 3.26. Albeit subtle, these colored glints are pervasively observed in the real world. To further verify this effect, we set up a simple scene in a dark room, illuminating a MacBook using an LED light from a cell phone. In this way, we can see strong colored highlights. Our method is able to produce perceptually similar appearance.

Tumbler. This scene illustrates a tumbler with brushed metal on the body under two point lights and environment lighting. The heightfield resembles brushing towards one direction, and thus is highly anisotropic. As shown in Figure 3.27, our method is able to handle the anisotropy, resulting in two vertical lines of highlights. From the insets, we can also see that the geometric method generates wider highlight peaks but narrower highlights overall, while our method is able to produce thin peaks but with much wider spread. This observation corresponds to the brushed BRDF images in Figure 3.23, where most energy concentrates in the central vertical line for wave optics. A similar observation was noted by Dong et al. [25]. Moreover, the observation is also in accordance with the geometric GGX BRDF [132], well known for its "long tail" and its ability to better represent slow falloffs of highlights than other geometric BRDFs such as Beckmann. The question of why wave optics often leads to longer tails is complex; a simple though incomplete explanation is that wave optics is less influenced by finer scale roughness, leading to sharper peaks, while off-

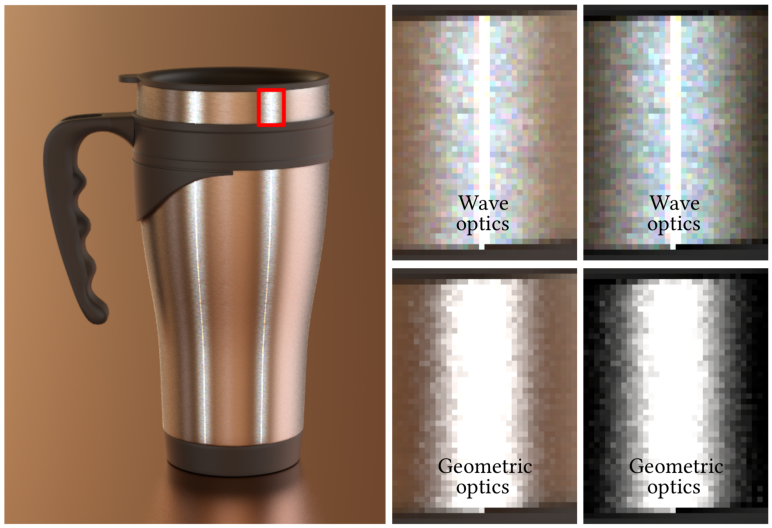


Figure 3.27: The Tumbler *scene rendered with two point lights and environment lighting, showing brushed aluminum with strong anisotropy. Insets compare our spectral method (top row) with the geometric method (bottom row). We can see that the geometric method produces wider highlight peaks but narrower highlights overall, and misses the colored glints.*

peak dropoff depends on destructive interference which tends to be somewhat random and incomplete, leading to stronger tails.

3.13 Summary

In this Chapter, we present detailed rendering for complex surfaces. We analyze the inability of statistical BRDFs, and come up with our $\mathcal{P}$-NDF solution on top of microfacet BRDFs. We first describe a basic approach to accurately calcuate $\mathcal{P}$-NDFs using triangulation, leading to closed-form solutions. Then we improve it by treating a detailed surface as a four-dimensional *position-normal distribution*, and fit this distribution using 4D Gaussian elements. This gives our method the ability of handling both evaluation and sampling queries, enabling the first practical solution with over 100 times speedup. Finally, we introduce wave optics to detailed rendering to correctly generate diffraction effects such as colors. We use Gabor kernels to approximate the phase shifts introduced by the underlying heightfield, and derive an analytic solution for spatically-varying, detailed wave BRDFs.

Our detailed rendering has changed the BRDF's nearly 4 decades of statistical use, but there is still room for improvements for our methods. One key issue is the storage of detailed surfaces, which may take up to the level of gigabytes for production. Efficient compression and querying thus become crucial to the practicality of our methods. Also, currently our approach only looks at single specular reflection, ignoring inter-reflection, layered materials,

refraction, or complex 3D structures (e.g., in biological iridescence). Furthermore, the computational expense of our wave optics approach still requires separation of direct illumination due to small lightsources from other components, and using a different (cheaper) BRDF for these components; further accelerations should be explored. Improved importance sampling techniques would also help. Physical measurement to acquire the heightfields would be another interesting addition.

With the development of display and computing technologies and the increasing demand for better graphics from people's everyday lives, we believe that detailed rendering is becoming a key for the next generation Computer Graphics and to achieve realism indistinguishable from the real world. Note that, our detailed rendering brings out the details from *known* microstructures. In the next Chapter, we will continue to another important component of photorealism: detailed appearance modeling, to deal with materials whose optical properties are *not yet fully understood*.

Chapter 4

Detailed Appearance Modeling of Animal Fur

4.1 Introduction

In Chapter 3, we have introduced detailed rendering. Given any specified microstructures on surfaces, our method is able to generate convincing details, thus enhancing the realism of computer synthesized images. Note that, in detailed rendering, how the surface material interacts with the light is known—millions of specular microfacets reflect rays like tiny mirrors.

Actually, it is exactly the optical behaviors of different materials that make it possible for us to distinguish whether certain materials are diffuse, glossy, or glinty. However, there are numerous kinds of natural materials in the real world where we are not exactly sure how they interact with the light. That explains why some of these materials, such as the butterfly wings, cloth and human skin, are still very difficult to render realistically.

Fortunately, in Computer Graphics, people study the optical properties of unknown materials or structures. This field is known as appearance modeling. In this chapter, we demonstrate the full process of appearance modeling from observations to accurate appearance/reflectance models. Specifically, we focus on physically-based modeling and rendering of animal fur.

In Computer Graphics, accurate physically-based fur rendering is often required for creating realistic furry appearance for animals. This is a long-standing problem [61] with many approaches proposed to address the geometric complexity of fur. However, current fur reflectance models are mostly derived empirically or from those for human hair, such as the Kajiya-Kay [61] or Marschner model [75]. Fur has a distinct diffusive and saturated appearance, as shown in Figure 4.1, which is not fully captured by these models since their focus is largely on the specular reflection and refraction. We also conduct measurements for a number of different types of fur, to confirm that previous hair reflectance models cannot fit reflectance profiles from fur fibers (see Figures. 4.7, 4.10, 4.11).

Figure 4.1: A rendering of the Wolf scene under environment lighting using (left) our physically-based double cylinder fur reflectance model with parameters from our database of animal fur samples, and (right) energy conserving Marschner model [23, 75] with best-fit parameters. Insets showing detailed comparisons from top to bottom using our model, Marschner model and Kajiya-Kay model. Since the Marschner model consists of only specular lobes, it often produces dark regions (limbs and tail). Furthermore, since the TT lobe is extremely strong in the Marschner model, especially for light colored fur fibers, it completely fails in heterogeneous regions (head) where dark colored fur is covered by light colored fur. The Kajiya-Kay model produces empirically plausible but hard-and-solid appearance, and it doesn't fit the measured reflectance data in Sec. 4.5.

These observations motivated us to look into the literature on differences between hair and fur [8, 18, 19] (Section 4.3). Briefly, a single fur fiber cannot be modeled as a simple dielectric cylinder similar to hair models. It often contains a non-negligible scattering structure inside called the medulla (Figure 4.2), which significantly affects the appearance of a fur fiber. In contrast, for human hair, the medulla usually takes up less than one-third of the fiber diameter and can be neglected. A key insight of this dissertation is to take medulla scattering into account for a novel physically-accurate fur reflectance model, leading to the following contributions:

Double Cylinder Fur Fiber Model: In Section 4.4, we develop the physical double cylinder model for a fur fiber. As shown in Figure 4.4, our model consists of three anatomically-based components—cuticle, cortex and medulla. The surface of the outer cylinder represents the cuticle, the inner cylinder represents the scattering medulla, and the cortex lies between them. We also introduce a multi-layer cuticle model to better capture reflection effects (Figure 4.3).

Measurement and Database of Fur Reflectance: To validate our physical model, we use a gantry setup to measure 2D reflectance profiles of single fur fibers from nine different animals, as well as a tenth measurement on human hair for reference (Section 4.5). We fit our physical model to the measured data, showing quantitative agreement in the space of 2D reflectance profiles (Figure 4.7). We also show that existing hair reflectance models like Kajiya-Kay and Marschner do not fit key features of the appearance (Figures 4.10, 4.11). These results clearly indicate that our accurate modeling of cuticle reflection and medulla

scattering are critical for fur rendering.

We provide the first database in computer graphics, of reflectance measurements and fit parameters for nine types of animal fur. The parameters can directly be plugged into our rendering model, or provide a baseline for an artist to slightly vary parameters to obtain different types of appearance. Fit parameters are listed in Table 4.2.

Next, we focus on the practicality of our double cylinder model, especially the medulla that results in complicated light paths. This complexity directly leads to significant pre-computation, and limits fur rendering to be near-field only. Even for hair rendering, efficient far field integration schemes are lacking. State of the art methods either assume that the azimuthal section of hair fibers are perfectly smooth [75] so that the far-field integration can be solved, or resort to numerical integrations as well as pre-computation [23].

Motivated by these observations, we aim to improve the efficiency and practicality of fur rendering, and provide a reflectance model that is simple to implement in modern rendering systems. Specifically, we describe a near field fur reflectance model in Section 4.8, focusing on simplicity and accuracy as compared to our original model. In Section 4.9, we illustrate how our near-field model integrates to far-field, and propose a novel multi-scale rendering scheme, focusing on efficiency. Overall, our major contributions are:

Simple reflectance model: Our local illumination model builds upon the double cylinder model representing the cuticle-cortex-medulla structure of fur fibers (Figure 4.18 (b)). We *unify the cortex and the medulla's indices of refraction (IORs)*, removing most of the complicated types of light interactions between them. This simplification finally results in only 5 lobes in our model (Figure 4.18 (c)), compared to 11 lobes previously. In particular, we keep the R, TT, and TRT lobes in hair models (with intensities modified slightly because of attenuation by the medulla) and only add two new scattered lobes, TT^s and TRT^s. In this way, our reflectance model (BCSDF) is fully analytic, as opposed to our basic model which requires implict ray tracing. The simplicity of our model also benefits importance sampling (Section 4.10), leading to faster convergence (Figures 4.28 and 4.29). Our BCSDF can be included in existing hair rendering systems with minimal extra effort, and unifies hair and fur rendering. We are motivated by observations in the literature on fur fibers (Figure 4.19), and we verify this simplification by the accuracy of fits to real measurements (Figure 4.24).

Improved accuracy and practicality: We improve the accuracy of the model by considering different roughnesses of azimuthal and longitudinal sections, as has been observed in previous studies [23, 10]. Moreover, we find that the medulla is not purely scattering, and can also absorb light [8]. With a more general model taking these factors into consideration, our error for fitting measured data is lower than previous in most cases, even with the simplification of IORs discussed above, as shown in Figure 4.24.

We also observed that the precomputed data for medulla scattering provided earlier, essentially a set of 4D tables, is low rank. We exploit tensor decomposition, the high dimensional analogue to 2D singular value decomposition (SVD), to compress it to as low as 150 KB, compared to more than 600 MB originally.

Analytic near/far field solution: Our model starts with analytic near-field solutions,

as opposed to the implicit ray tracing required previously. Then we integrate our near-field model over the azimuthal section, by partitioning the range of integration into a few (< 5) segments. Finally, we analytically integrate for each segment using piecewise linear approximation. Moreover, we show how our analytic integration transitions between near and far field fur rendering, enabling multi-scale rendering for the first time (Figure 4.30). This is especially useful when a pixel covers a small range over the azimuthal section. Our multi-scale rendering benefits hair rendering as well, as shown in Figure 4.32.

Significant speedup: Due to the simplicity of our reflectance model and the efficiency of our analytic integration, compared with our original double cylinder model, we achieve a $6 - 8\times$ speedup in generating equal quality results. We show more results and comparisons in Section 4.10 as well as in the accompanying video.

With our improved near and far field model, physically-based hair and fur rendering is now possible. However, accurate rendering also requires global illumination for the diffusive and saturated appearance of the hair or fur volume. In fact, compared to local illumination that considers how light interacts inside individual fibers, global illumination caused by light scattering within the fur volume is usually brighter and more visible, composing the main part of the appearance.

However, rendering with global illumination is slow. Therefore approximate methods are common, the most popular of which is the dual scattering technique [154]. Dual scattering produces reasonably good results and allows real-time implementation[1]. It simplifies light scattering by assuming that all scattering events happen only along *main paths*, i.e. straight lines along incident directions. However, it is not applicable for fur rendering. This is mainly because the complex scattered lobes within individual fur fibers break the main path assumption. Even if we extend dual scattering to handle these scattered lobes, the result still doesn't match the path traced reference (Figure 4.33). This will be analyzed in detail in Section 4.11. Moreover, dual scattering only works with direct lighting, and is limited to point lights and directional lights. It also does not support transparency of hair fibers. Compared to path traced results, dual scattering still generates a hard and solid appearance (Figure 4.33).

We develop a novel BSSRDF (Bidirectional Surface Scattering Reflectance Distribution Function) solution to global illumination, addressing many of dual scattering's limitations. We analyze failure cases of dual scattering in Section 4.11, and propose our model with three components: direct illumination from individual fur fibers, dual scattering to handle specular light transport, and BSSRDF for all other scattering events. In Section 4.12 and 4.13, we describe our BSSRDF model, and explain how to convert properties from fur fibers to BSSRDF parameters with the help of a multi-layer perceptron neural network (MLPNN). We validate our model in Section 4.14, and show close matches of our predicted renderings, compared with the path traced reference.

[1]Currently, we focus on the offline rendering part of dual scattering as well as our method, though real-time implementation is a natural next step for our model.

Specifically, our global illumination model has these major advantages:

BSSRDF-based approximate global illumination. Our method is the first approximate global illumination model that is suitable for both hair and fur rendering. We also provide the first empirical scheme of conversion from hair/fur parameters to BSSRDF parameters using a neural network. Our neural network is simple, consisting of only two hidden layers, and is fast to evaluate and easy to integrate into renderers for practical use. Furthermore, it only uses one scene for training with different parameters, and it generalizes well on others.

Color bleeding and accurate appearance. Dual scattering and other non-physically-based methods assume light transport only along main paths and assume local similarity with the fur fiber being shaded, resulting in opaque solid colors and BRDF style global illumination approximation. Our BSSRDF model is able to handle color bleeding from the fur volume for the first time, e.g. the color-filled shadow in Figure 4.33 (c), and its general softer appearance. Moreover, in Section 4.14, we show that our model also generates much more accurate appearance, especially in terms of highlight and overall shading distributions, compared to the reference.

Further inter-reflections. Dual scattering works only with direct illumination. It is not clear how to apply it in offline renderers where environment or indirect lighting also lights the hair volume, and where the lit hair volume affects other objects. In contrast, our BSSRDF global illumination model (Section 4.11) naturally fits into offline renderers with environment lighting and further global illumination (Section 4.13). Moreover, we extend our BSSRDF model to handle global fur-to-fur inter-reflections approximately for the first time.

Efficient performance. In addition to its accuracy, our method is also efficient. Thanks to the BSSRDF's ability to "sum up" complex scattering events and higher order bounces, we're usually able to achieve an order of magnitude speedup, compared with the path traced reference with our improved double cylinder model. Detailed timing information is listed in Figure 4.33 and Section 4.14.

4.2 Related Work

Physically-based hair reflectance models: Marschner et al. [75] proposed a physically-based hair reflectance model. The hair fibers are considered as rough dielectric cylinders, where three scattering paths contributing to the primary and secondary highlights are modeled: R, TT and TRT. The reflectance lobe from each path is separated into a product of longitudinal and azimuthal scattering profiles. Zinke et al. [152] formalized hair reflectance models by introducing the notion of the Bidirectional Curve Scattering Distribution Function (BCSDF). Sadeghi et al. [107] reformulated the model of [75] into an artist friendly representation. d'Eon et al. [23] extended Marschner's model from an energy conserving perspective by modeling higher-order scattering lobes such as $TRRT$, and by fixing energy-

conserving issues at grazing angles. Recently, d'Eon et al. [21] proposed a "non-separable" reflectance lobe representation by relating longitudinal contributions with relative azimuths, while still keeping a factored representation longitudinally and azimuthally. These methods produce excellent results for hair, but are not suitable for fur (see Figures 3.1, 4.10), since they exclude scattering from the medulla, which is prominent in fur fibers. Concurrent work [63] focuses on elliptical hair fibers explicitly, revealing different optical properties compared to circular sections.

Non physically-based hair/fur reflectance models: Kajiya and Kay [61] introduced a methodology for rendering fur using 3D textures, together with an empirical fur shading model. The model approximates fur fibers as opaque cylinders. By extending the Phong model, it produces a diffuse lobe and a specular lobe centered around the fiber's tangent. Goldman et al. [36] empirically improved the Kajiya-Kay model by giving it different opacity values for different viewing angles. Zinke et al. [153] noticed the inability to fit the measured scattering from human hair fibers using Marschner's model, so they proposed an ad-hoc method by blending a diffuse lobe with Marschner's model to capture the diffusive hair reflectance observed in their measurement data. Though these methods generate plausible rendering results, they are not physically based, nor energy conserving. Moreover, they do not fit the observed reflectance profiles for fur fibers as accurately as our physically-based model.

Multiple scattering inside hair volume: Since hair contains many fibers, multiple scattering is difficult to compute. The dual scattering approximation [154] assumes local similarity of hair strands, and derives an analytical multiple scattering model, which is later extended by [140] to enable real-time rendering and editing under environment lighting. Shadow map related methods [73, 151, 110] use transparency to compensate the missing transmittance of light in [61]. Moon et al. [83] applies photon mapping into the hair volume, trading noise with overblur or bias. In this dissertation, we focus on the optical properties from a single fur fiber, and use a standard multiple scattering renderer [51] to obtain global effects.

Importance sampling for hair: To be efficient, a reflectance model requires an importance sampling method in a global illumination renderer. Hery and Ramamoorthi [47] proposed an importance sampling scheme for reflectance lobes of the Marschner model. Ou et al. [90] extended the sampling scheme to separately sample different lobes. d'Eon et al. [22] proposed an efficient technique based on the extended model from [23] by first performing a lobe selection based on the energy of each scatter type, then importance sampling the longitudinal and azimuthal scattering profiles respectively. We develop an effective importance sampling method for our fur reflectance model, based upon [22].

Near field scattering and far field approximation: Far-field approximation based methods [61, 75, 23] regard single hair fibers as thin curves, and assume collimated incident light rays over the width of the fiber. This approximation gives hair fibers a flat appearance at a close viewing distance. Zinke et al. [152] proposed an analytical near-field solution to render fibers viewed up close. Our method is based on near-field scattering, and we refer to

the same integration technique in [23] when far-field approximation is needed.

Precomputation and empirical models: Precomputation-based rendering methods such as [113, 135, 86, 87, 99] work by solving a subset of the problem in advance. These methods are usually very efficient. However, since precomputed data can take up significant storage, they are practical only under a confined set of inputs. Empirical models such as [117, 93, 26] use approximations and numerical methods to get the reflectance or scattering profiles. These models often require measurements or simulations covering the entire parameter space. Our model includes a precomputation step in a low-dimensional 3D parameter space to account for medulla scattering, storing a 1D profile for each combination of these parameters. We compress the precomputed profiles to 20MB.

Near field scattering and far field approximation: Far-field approximation [61, 75, 23] assumes that hair fibers are very thin, usually thinner than a pixel. Hence, the accurate incident position on the azimuthal section is not important, compared to the integral over it. Thus, they always assume collimated incident light covering the width of a hair fiber. Since the positional information is lost, far field approximation produces a flat appearance when viewed close up so that a hair fiber covers more than one pixel. Zinke et al. [152] introduced near-field scattering by considering accurate azimuthal incident positions, and mathematically revealed the relationship between near and far field scattering. We propose an accurate and analytic method for our improved model with support for both near-field and far-field rendering, benefiting both hair and fur models.

Hair global illumination methods: Accurate global illumination requires simulating actual light bouncing between hair fibers. Moon et al. [83] extended photon mapping to store and query photons within the hair volume as light bounces inside. Hery et al. [47] and d'Eon et al. [22] proposed different importance sampling schemes for hair BCSDFs to accelerate the convergence of path traced global illumination.

Most non physically-based approximate global illumination methods treat hair fibers as semi-transparent. To compensate for light transmittance through fibers to the shading point, shadow map based methods [73, 151, 110] accumulated transparency of hair fibers to approximate the optical thickness of hair fibers that the light goes through. To enable transparency looking from the camera, alpha blending based methods [111, 33, 110, 150] proposed different ways to approximate the back-to-front blended self-occlusion effects. While generating plausible results, none of these methods are physically-based, and they have not been shown to be applicable for accurate fur reflectance models.

The only physically-based approximation to hair global illumination is the dual scattering approximation method [154]. It assumes that scattering events happen along the main path — the light reaches the shading point by penetrating through the hair volume in a straight line (global scattering), and scatters along the camera path into the hair volume, then back to the shading point (local scattering). Though successfully used in practice, dual scattering cannot be applied for fur rendering (Section 4.11). Furthermore, it doesn't account for transparency or color bleeding.

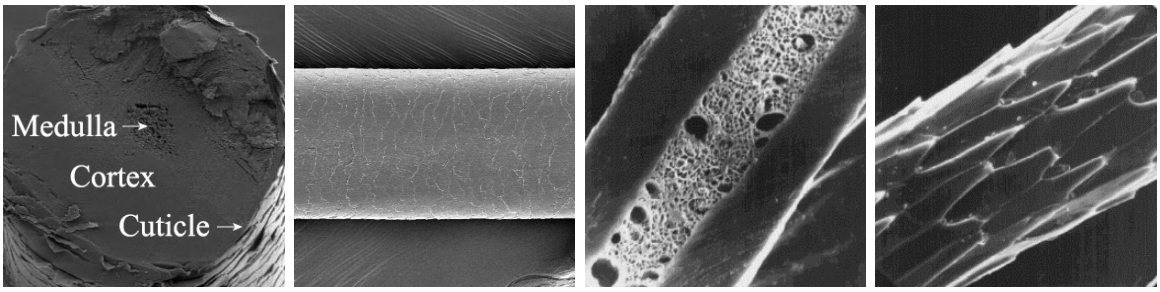


Figure 4.2: Structure of human hair and animal fur fibers. From left to right: section of a human hair fiber, cuticle of a human hair fiber, section of a cougar fur fiber, cuticle of a corsac fox fur fiber. Note major differences in the size of the medulla and complexity of the cuticle. Images authorized by [137, 34].

4.3 Differences between Hair and Fur Fibers

In this section, we describe key differences between hair and fur fibers. While the subject has received little attention in computer graphics, a number of references in other fields [8, 18, 19, 118] discuss microscopic variations, that we summarize here.

Hair and fur fibers share some common structures. They are often cylindrical shaped with some extent of eccentricity. As shown in Fig. 4.2, from outer to inner, a single fiber is divided into three layers: the *cuticle* which covers the fiber's surface with inclined scales, the *cortex* which contains nearly all colored pigments within the fiber, and the *medulla* which lies in the center of the fiber with complex internal structure that scatters light that goes through.

Inspite of these common structures, hair and fur fibers do have several structural differences. Here we only introduce the most important features for our model that result in clearly different optical properties. For a comprehensive and detailed study, we refer the reader to the literature mentioned at the beginning of this section.

Medulla: The most obvious difference is that animal fur fibers usually have significant medullas inside. For human hair, the medulla is very small, and it can often be neglected. However, animal fur fibers can have medullas that hold up to the total size of the cylinder (Fig. 4.2).

The structure within the medulla volume is often complicated, while some animals, such as polar bears, have hollow medullas in their fur. In any case, the medulla acts as an internal scattering structure, giving the fur a generally diffusive appearance. The medulla could be filled with solid transparent materials or simply air, which indicates that the medulla could have a different refractive index compared to the cortex. There are usually no pigments inside the medulla.

Cuticle: While a human hair shaft has cuticle scales that resemble roof shingles, cuticles on fur can have complex shapes (Fig. 4.2). The outer surface of animal fur fibers is usually rougher than that of human hair. Following [23], we account for roughness of both longitudinal and azimuthal sections of a fur fiber in our model, assuming they have the same

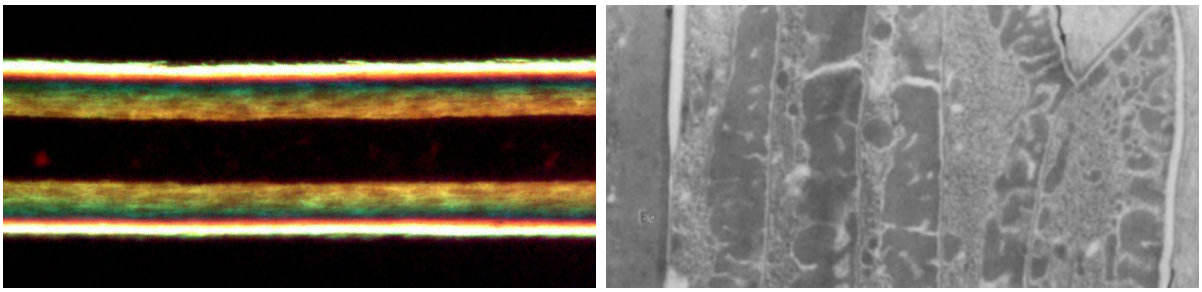


Figure 4.3: (Left) A microscope photograph of a sample of polar bear fur. Note the obvious coating formed by the cuticle scales. (Right) A slice of cuticle scales on human hair shaft. Images authorized by [8, 44].

value for simplicity.

Additionally, as Fig. 4.3 shows, the cuticle layer forms a clear coat over the cortex, within which, multiple cuticle scales stack up and form a layered structure. These properties increase cuticle reflectance compared to Fresnel reflectance from a dielectric interface. In our model, we consider the cuticle as multiple layers of dielectric slab with air outside both sides of each layer. The unpolarized reflectance for each layer is given by [119] as

$$\begin{aligned} F(\theta_i, 1) =& \frac{1}{2}\left\{ F_s(\theta_i) + \frac{\left[1 - F_s(\theta_i)\right]^2 F_s(\theta_i)}{1 - F_s^2(\theta_i)} \right\} + \\ & \frac{1}{2}\left\{ F_p(\theta_i) + \frac{\left[1 - F_p(\theta_i)\right]^2 F_p(\theta_i)}{1 - F_p^2(\theta_i)} \right\} \end{aligned} \tag{4.1}$$

where F_s and F_p are s-polarized and p-polarized Fresnel reflectance respectively. Considering l layers together, the reflectance is then given by [119] as

$$F(\theta_i, l) = \frac{l \cdot F(\theta_i, 1)}{1 + (l - 1) \cdot F(\theta_i, 1)} \tag{4.2}$$

As pointed out by [118] and [44], the internal composition of the cuticle layer may give rise to a different refractive index than the cortex. Additionally, the air interface between layers could be absent. We simplify all these properties by extending l into a real number, while using the same refractive indices for the cuticle and the cortex. The value of l is decided in our fitting step in Sec. 4.5. l is usually within a typical range of $(0.5, 1.4)$, while a single dielectric interface produces a value of $l \approx 0.5$, since $F(\theta_i, 1)$ is for a double-sided slab.

4.4 Double Cylinder Fur Fiber Model

We now develop our physical double cylinder model for single fur fibers, based on the above observations. Our model consists of three structural components—cuticle, cortex

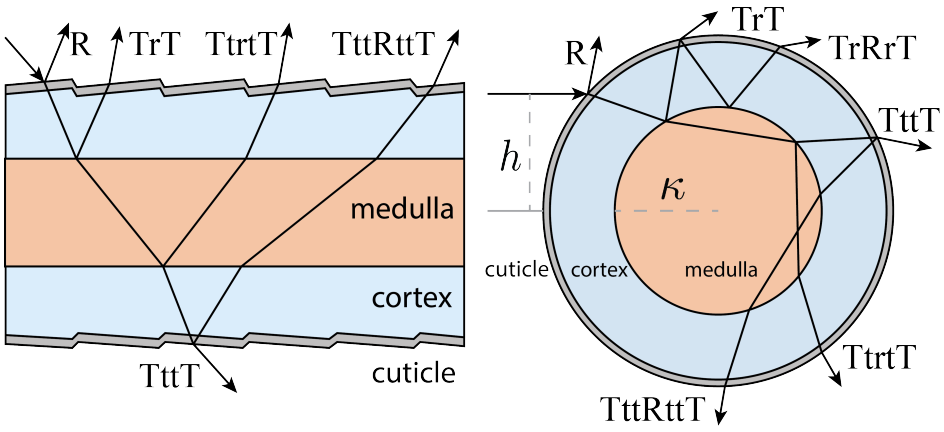


Figure 4.4: Schematic of our double cylinder model in longitudinal section (left) and azimuthal section (right). Our model considers the medulla and cuticle effects as introduced in Sec. 4.3. We mark new types of paths TrT, $TrRrT$, $TttT$, $TtrtT$, $TttRttT$ that our model introduces. For clarity we hide TT and TRT paths that were previously considered by [75] in Fig. 2.6. These TT, TRT paths enter the cortex, but miss the medulla entirely.

and medulla — each with their respective physically-based optical properties. As shown in Fig. 4.4, the surface of the outer cylinder represents the (multi-layer) cuticle, the inner cylinder represents the medulla, and the cortex lies between them. The two cylinders are coaxial with relative radius 1 for the outer cylinder and κ for the inner cylinder, which is known as the *medullary index*, i.e., the ratio between the radius of the medulla and the radius of the fur fiber.

Table 4.1 lists all the parameters used in our model. The parameters derive largely from those in [75]. These include the refractive index η and roughness β, the scale tilt α and absorption coefficient $\sigma_{c,a}$ for the outer cylinder (cortex). In addition, we need to consider the (probably different) refractive indices in the cortex and medulla. Moreover, since the medulla is a scattering medium, we must include its scattering coefficient $\sigma_{m,s}$ and its phase function with anisotropy g. Note that the cortex doesn't have scattering structures inside it, and the pigments are seldom found in the medulla, as stated in Sec. 4.3. Hence, the scattering of the cortex and the absorption of the medulla are not required in our model. We don't explicitly use the eccentricity parameter, but it is taken into account in the refractive indices, as in [75].

With the double cylinder model, the types of paths through a single fiber are significantly enriched. Apart from those that were previously considered in hair models, our model introduces new types of paths that are not captured by previous methods such as TrT, $TrRrT$, $TttT$, $TtrtT$ and $TttRttT$, as shown in Fig. 4.4. The upper-case letters indicate interactions with the outer cylinder interface, while the lower-case ones are with the inner cylinder interface. When light transmits through the inner cylinder interface, volumetric scattering events happen.

The medulla's significant contribution to the optical properties of a single fiber could be

Parameter	Definition
κ	medullary index (rel. radius length)
η_c	refractive index of cortex
η_m	refractive index of medulla
α	scale tilt for cuticle
β	roughness of cuticle (stdev.)
$\sigma_{c,a}$	absorption coefficient in cortex
$\sigma_{m,s}$	scattering coefficient in medulla
g	anisotropy factor of scattering in medulla
l	layers of cuticle

Table 4.1: Parameters used in our double cylinder model.

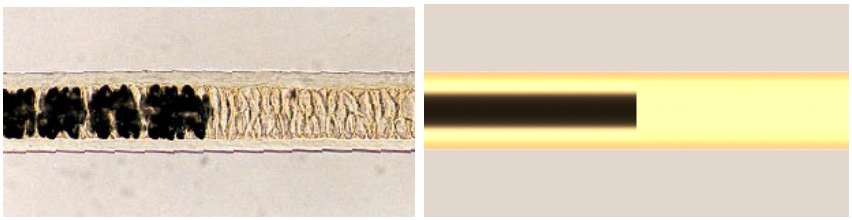


Figure 4.5: (Left) A photograph of a real fur fiber under bright field microscopy with medulla filled half with air and half with a mounting medium. (Right) Our Monte Carlo simulated back-lit microscopic appearance of fur fiber samples with unmounted and mounted medulla (two images stitched). Photograph publicly licensed by [18].

observed when the medulla is filled with a medium of similar refractive index as the cortex. This makes the medulla much more homogeneous and significantly reduces scattering. As shown in Fig. 4.5, when it is back-lit, the medulla filled with mounting medium appears nearly invisible. However, if filled with air, it is dark due to internal reflections and scattering. Our model produces similar results by tuning $\sigma_{m,s}$ and η_m accordingly. To our knowledge, this phenomenon cannot be simulated with any other hair/fur reflectance models.

4.5 Database: Measurements and Validation

To justify the double cylinder model, and to experimentally observe the influence of the cuticle and the medulla, we take full 2D far-field reflectance measurements of a fur fiber. We create a database of reflectance profiles using fur fibers of 9 animal species (plus a human hair). From each dataset, we also fit a set of parameters for the double cylinder model. The raw reflectance data and the fit parameters are available on `http://viscomp.ucsd.edu/projects/fur`.

Measurement Setup: Measurements are made using the UCSD spherical gantry (Figure 4.6). Our gantry has two robotic arms, on which the light source and the sensor are

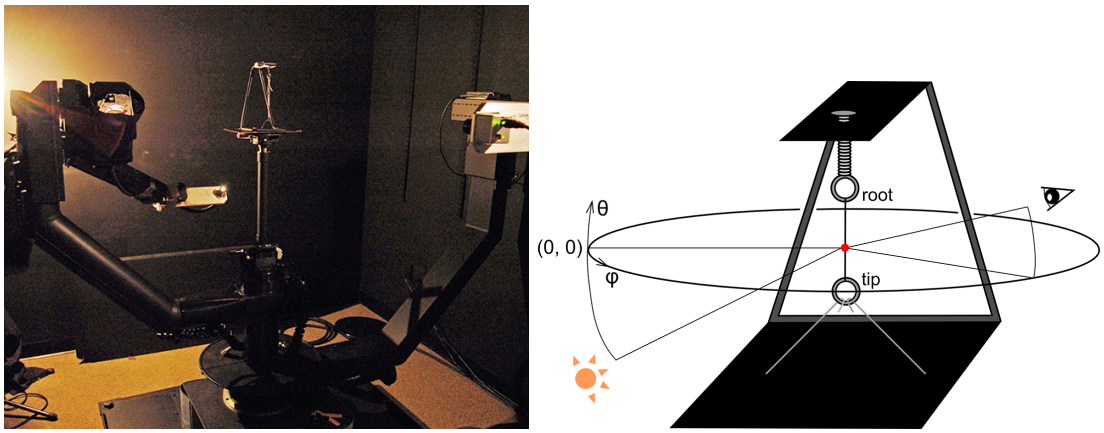


Figure 4.6: (Left) The spherical gantry we use to measure individual fibers' reflectance profiles. (Right) Illustration of setup for two-dimensional far-field reflectance measurements.

attached at approximately 2 feet and 3 feet away from the sample respectively. We use a 150-watt DC-regulated quartz halogen bulb, a digital camera with 35 mm lens and a 12-bit 1/1.8" CCD sensor. We straighten a fur fiber on the sample plate, fix the incident direction of light at a point on the fiber, and record radiance towards discretized directions over the entire outgoing sphere. More details are found in [125].

We fix the light source at $(\theta_i, \phi_i) = (-40°, 0°)$ and capture images with the camera sweeping over $\phi_r \in [-20°, 200°]$ at 5° steps and $\theta_r \in [10°, 50°]$ at 2° steps. For simplicity, we omit the r subscripts and just use (θ, ϕ) to describe the 2D reflectance profiles. The swept range covers all the specular and diffusive scattering lobes introduced in prior human hair reflectance models, but only spans across a quarter-sphere for efficiency reasons. In other words, we implicitly assume that the fur fiber is symmetric over the incident plane. To ensure the validity of this assumption, before taking 2D measurements, we spin each fur fiber along its tangent until a qualitatively symmetric 1D normal plane reflectance profile is captured. If such symmetry in reflectance pattern is never observed, we simply dispose of the sample. The proportion of samples disposed due to asymmetry is about 10%.

For each direction, we capture 5 images for 5 stops in shutter speed (25 images in total). Each image is first cropped to a 25-by-25-pixel patch which contains the fiber. Next, the 5 patches for each shutter speed are averaged to eliminate temporal noise. Finally, we leverage the 5 averaged patches under different shutter speeds to reconstruct an HDR radiance signal. During this process, the aperture of the camera is fixed at f/8, and the 5 stops in shutter speed to construct HDR radiance values are 12.5, 25, 50, 100 and 200 ms. In each of the 25-by-25-pixel image patches, the sample fiber takes up approximately 100 to 200 pixels, depending on its actual width. Note that, for measurements and subsequent fits, we consider the data as gray-scale images.

Database: The 9 animal species in the database are bobcat, cat, deer, dog, mouse, rabbit, raccoon, red fox and springbok. Fur fibers are donated by a taxidermy store. We also measure the reflectance profile of a human hair for comparison and verification against prior

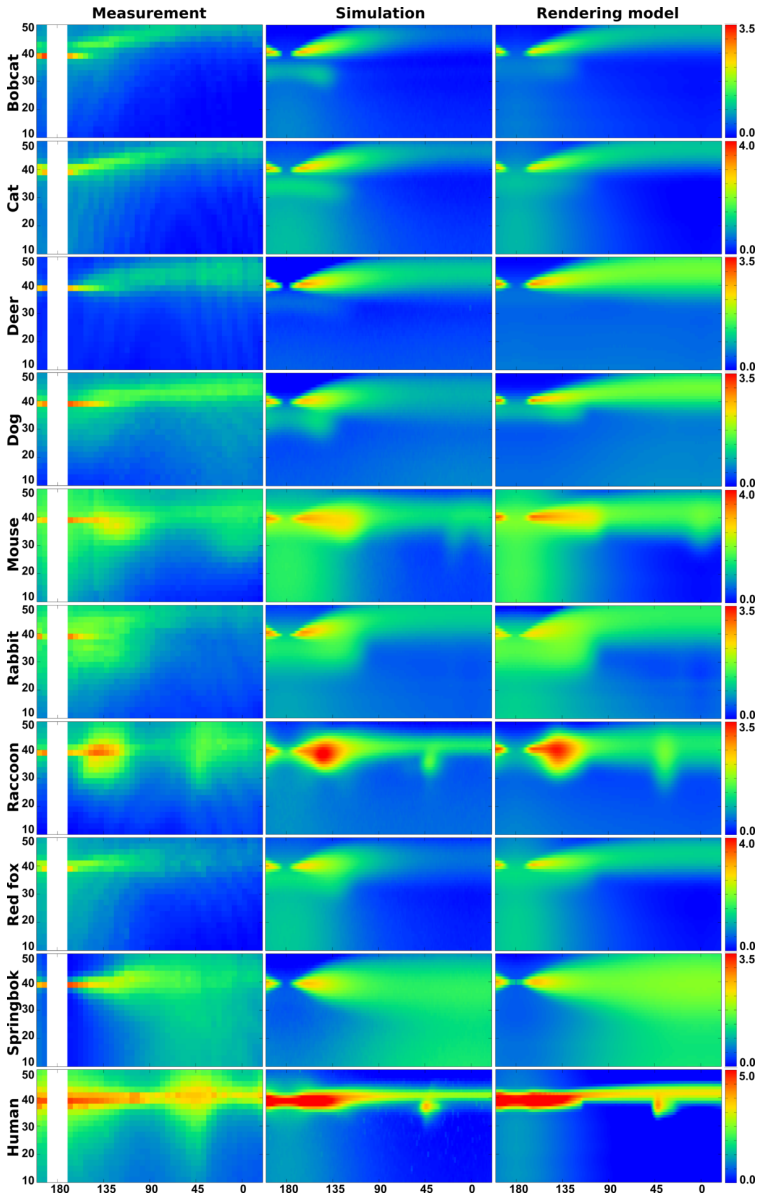


Figure 4.7: 2D reflectance profiles measured from different animals' fur fibers (left), synthesized from full 3D volumetric path tracing of a double cylinder (middle), and from our factored rendering model from Section 4.6 (right). The signals are in arbitrary units and displayed in logarithmic space to visualize perceptual brightness.

work. Our goal is to investigate the range of reflectance patterns from fur fibers. We do not focus on taxonomic details; the names of the 9 species are only for reference. The recorded profiles show that the 10 samples (including a human hair fiber) do span a qualitatively large space in the BCSDF domain, and we consider them to include the most important reflectance phenomena.

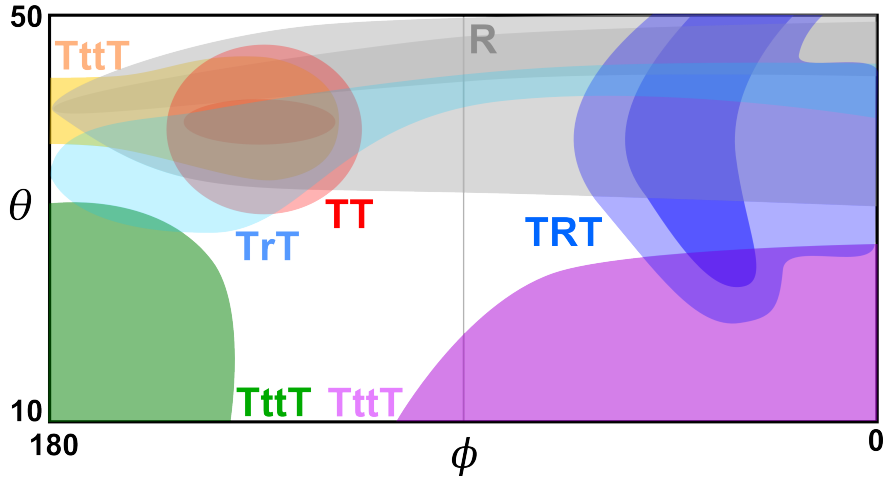


Figure 4.8: Illustration of general positions and shapes of all the major lobes we observed in our database.

The entire database is displayed in the left column of Figure 4.7. Note that a small region of the 2D profile around $\phi = 180°$ could not be measured accurately, where the camera points directly at the light source[2]. Also note that many profiles resolve subtle patterns of bright and dark stripes. We believe that these stripes arise from thin-film interference between light reflected off the front and the back side of the cuticle layer.[3] Modeling these interference patterns in our double cylinder model is beyond the scope of the dissertation.

A schematic of the key lobes observed in these measurements is shown in Figure 4.8, and some examples are given in the left column of Figure 4.10. The name for each lobe corresponds to its contributor path, where their assignments are verified through virtual experiments using volumetric light transport simulation on 3D double cylinder models. The Marschner model fails to produce satisfying fits to the reflectance profiles of fur fibers, as shown in the right column of Figure 4.10, since it supports only the R, TT and TRT lobes. Furthermore, even for these 3 lobes, there exist significant inconsistencies between what the Marschner model predicts and what we observe in the reflectance profiles. Some key features are:

- The reflectance from a fur fiber may have a blurry and partly occluded TT component around $\phi = 180°$ (mouse and raccoon) or even an absent TT component (cat). Note especially the sharp edge at $\phi \approx 150°$ where TT vanishes for the raccoon. Additionally,

[2]Since the light cannot be a perfect point light, a single fur fiber could not fully block it, thus resulting in extremely bright values. In practice, we narrow the range of the light using four paper-made walls around it, reducing the unmeasurable range to at most 10 degrees azimuthally.

[3] We check the difference in incident angle between rays that contribute to two consecutive bright fringes on our profiles. An analysis in optical path difference shows that to resolve the pattern requires the thickness of the cuticle to be approximately 10 micrometers, which conforms in magnitude with values reported in the literature.

the TRT component may become dimmed (mouse) or absent (cat). These phenomena are due to light being scattered away from the original TT and TRT paths by the medulla.

- The R component in the reflectance profile of a fur fiber is usually extremely bright and blurry, and cannot be explained with Fresnel reflection alone (dog, mouse, raccoon and springbok). The phenomena are caused by the cuticle, where the reflectance is boosted by reflection off the front and the back surfaces of the cuticle.

- Forward and/or backward diffusive lobes appear in many profiles. The centers of the lobes mostly lie around the normal plane where $\theta \approx 0^\circ$ (cat, red fox, springbok). These lobes are most likely due to the medulla, which provides a volume filled with randomly distributed dielectric interfaces and is prone to multiple scattering. Additionally, less-scattered light through the medulla can still be observed in some cases as a glow around $(\theta, \phi) = (40^\circ, 180^\circ)$ (mouse, red fox).

All of these phenomena can be well modeled by the double cylinder model proposed in the previous section (middle column of Figure 4.7).

Parameters and Fitting: We now fit parameter values (Table 4.1) for our double cylinder model from the measured 2D reflectance profiles. We can then use these parameters to define an explicit double cylinder geometry, and run a 3D volumetric light transport simulation. We compare the simulated reflectance profiles to the measurements (second column of Figure 4.7). We also compare (third column) to the reflectance generated by our factored rendering model introduced in Section 4.6 with the same set of parameters.

As our model has more parameters than previous hair models, fitting parameters over 1D slices of the measured reflectance profiles as in [75, 106] would lead to over-fitting. We therefore fit directly to the measured 2D profiles.

For each fur sample, we initialize a first estimate of the parameters manually. Then, we run full 3D volumetric light transport simulation over the double cylinder to generate a synthetic 2D reflectance profile. We use an expectation-maximization algorithm to iteratively find the optimized parameters with lowest root-mean-squared error between the measured and simulated 2D reflectance signals. This optimization is quite expensive (but need only be done once offline), since simulation must be used in the inner loop of the optimization to generate simulated reflectance profiles and must be noise-free so as to provide a smooth energy landscape. However, at the resolutions of our measured reflectance profile (45×21) and using 512 samples per pixel, the optimization procedure can converge to an acceptable minimum within several hours. A similar optimization is used to fit parameters of the Marschner and Kajiya-Kay models, where needed for comparison, with the only difference being that we directly use the analytical form of these models to generate 2D profiles, instead of using Monte Carlo simulation.

Table 4.2 lists the optimized parameters in our double cylinder model for each fur sample. We also render a fur ball for each material in Figure 4.9 using our practical rendering model

Parameter	Unit	Bobcat	Cat	Deer	Dog	Mouse
κ	unitless	0.78	0.85	0.87	0.69	0.60
η_c	unitless	1.40	1.43	1.54	1.55	1.38
η_m	unitless	1.27	1.35	1.42	1.37	1.38
α	degree	4.44	3.97	2.93	2.47	1.05
β	degree	4.86	4.94	5.35	4.21	4.70
$\sigma_{c,a}$	diameter^{-1}	0.75	0.48	1.81	0.37	0.50
$\sigma_{m,s}$	diameter^{-1}	3.18	2.58	2.75	3.17	2.93
g	unitless	0.54	0.62	0.39	0.18	0.65
l	unitless	0.50	0.59	0.69	0.53	0.89
Simulation NRMSE		8.1%	6.7%	8.4%	9.1%	7.8%
Rendering model NRMSE		7.2%	5.3%	7.9%	9.1%	8.5%

Parameter	Unit	Rabbit	Raccoon	Red fox	Springbok	Human
κ	unitless	0.66	0.59	0.69	0.85	0.34
η_c	unitless	1.36	1.23	1.43	1.55	1.21
η_m	unitless	1.34	1.23	1.38	1.32	1.21
α	degree	4.41	1.20	2.25	0.03	0.87
β	degree	6.97	5.27	4.86	8.43	2.03
$\sigma_{c,a}$	diameter^{-1}	0.83	0.38	0.73	0.96	0.83
$\sigma_{m,s}$	diameter^{-1}	2.53	3.45	2.99	3.06	4.30
g	unitless	0.31	0.35	0.63	0.03	0.38
l	unitless	0.65	1.51	0.53	0.54	1.49
Simulation NRMSE		9.4%	12.2%	5.9%	8.4%	15.4%
Rendering model NRMSE		8.4%	10.1%	6.3%	7.0%	19.3%

Table 4.2: (Top) Optimized parameters fit from our measured data using our simulated double cylinder model. Our rendering model shares exactly the same set of parameters. All length-related parameters are calculated assuming the azimuthal section of every fiber is a unit circle. All angle-related parameters are in degrees. (Bottom) Normalized RMS error of our simulated model and our rendering model.

that will be introduced in Section 4.6. Note that, since our measurements are gray-scale, we need to set the RGB values for the cortex absorption manually to introduce color in the renderings. For all the fur balls, we set $\sigma_{c,a} = 0.2, 0.4, 0.6$, replacing the original fit parameter $\sigma_{c,a}$.

Several observations can be made from Figure 4.9. First, none of these renderings display large black regions. This is difficult for Marschner model to replicate, since its lobes are all purely specular. Figure 4.17 further validates this observation, showing that the Marschner model produces large black areas under area lighting. Second, forward/backward scattered lobes affect the appearance significantly. Strong forward scattering (mouse, red fox) blurs regions in the middle of these fur balls, where there are usually more fur fibers behind, so

Figure 4.9: Renderings of a fur ball with 9 different sets of fit parameters. All the images are rendered using 1024 samples per pixel with top-front area lighting.

that the forward scattering energy penetrates and further scatters. Backward scattering (springbok) blurs the top and bottom regions, where the fur layer is usually thin and not viewed perpendicularly, and it visually enhances the reflection lobe. These effects indicate that the Kajiya-Kay model is not enough to represent the scattered lobe, since it is strictly uniform azimuthally, as shown in the comparison in Figure 4.11. Third, for reflection lobes with similar roughness and cortex refractive indices, the one with more cuticle layers l always produces stronger reflection lobes (cat slightly brighter than bobcat, deer much brighter than dog). Thus, both our medulla and cuticle models make a qualitative difference in the appearance of fur, which cannot be captured by previous methods.

Validation: As shown in Figure 4.7, we demonstrate quantitative similarities between the measured 2D profiles, simulated ones by path tracing an actual double cylinder, and

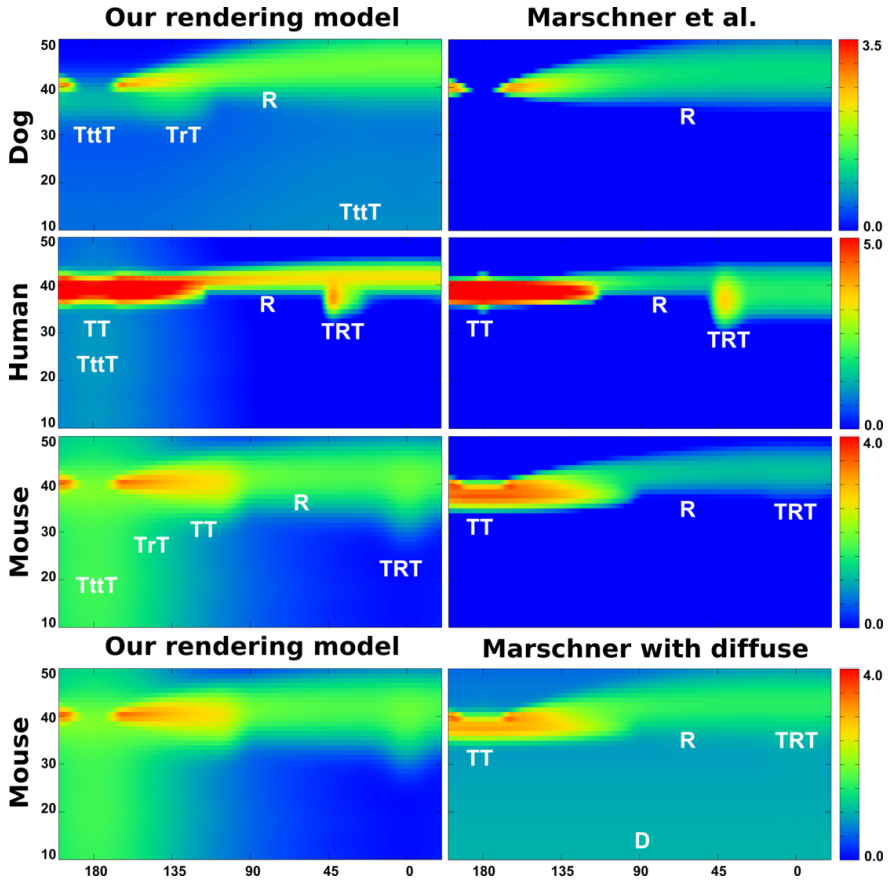


Figure 4.10: Comparisons of synthesized profiles using our rendering model (left column) and Marschner model (right column, first three rows). We also compare with empirically synthesized profiles by blending a diffusive (D) lobe into the Marschner model as in [153] (right column, last row). From top to bottom: dog fur profiles, human hair profiles, mouse fur profiles. Lobes are marked in the synthetic 2D profiles.

profiles generated using our rendering model,[4] which will be introduced in Section 4.6. The RMS errors are shown in Table 4.2, and are comparable for the simulation and rendering model, being below 10% in almost all cases.

We compare visually with the Marschner model in Figure 4.10. Since the Marschner model is purely specular, it fails in every measured profile to produce similar results, except in the case of human hair. Even when a diffuse lobe is blended into it, as suggested by [153], it still cannot match the measured reflectance. This is because the blended-in diffuse lobe is always symmetric longitudinally at $\theta = 0°$ and doesn't distinguish forward and backward scattering effects. We also show comparisons for 2D profiles generated using the Kajiya-Kay model. Since the Kajiya-Kay model is azimuthally independent and its shape changes slowly,

[4]Since our rendering model uses near field scattering, we integrate over azimuthal offset h to generate these far field 2D profiles, using the same technique as that introduced by [23].

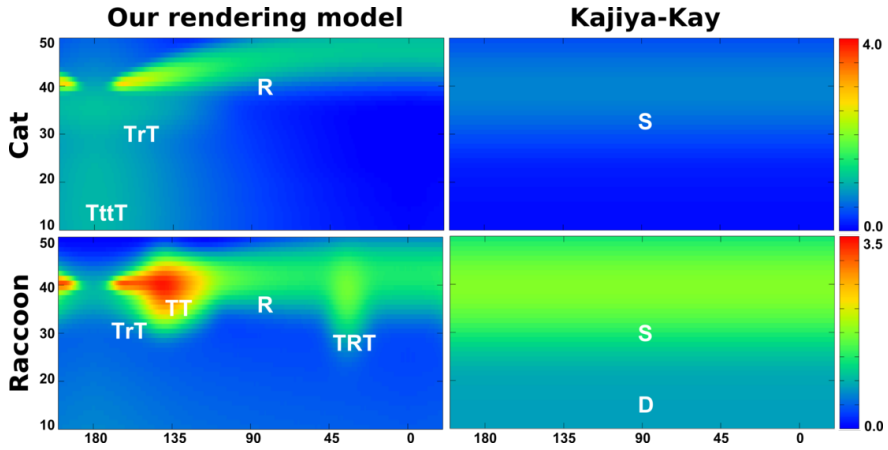


Figure 4.11: Comparisons of synthesized profiles using our rendering model (left column) and the Kajiya-Kay model with specular (S) and diffuse (D) lobes (right column).

we only compare two representative fur fibers in Figure 4.11. Kajiya-Kay produces only a diffuse lobe and a specular cone, eliminating all other interesting lobes.

4.6 A Practical Rendering Model

The physical double cylinder model enables us to run light transport simulation on an optically faithful representations of a fur fiber. However, as evaluating integrals over multiple volumetric scattering events is costly, the model remains impractical for rendering. In this section, we develop a practical rendering model, which can be plugged into global illumination renderers.

Overview

Under the far-field approximation, most prior reflectance models for hair obtain a compact BCSDF for the fiber. However, as introduced in Section 4.2, to model more sophisticated appearances of fur at a close distance, it is essential to include near-field effects; if necessary, these could be integrated to obtain far-field approximation. Thus, we analyze the potential scattering paths for an incoming ray hitting the double cylinder, based on its incident position and direction.

We make this approach practical through a few key approximations. First, our near-field model incorporates factored approximation for rendering, similar to far-field hair models [75, 23]. However, unlike those models, which consider a longitudinal and an azimuthal reflectance profile being generated collectively by all collimated incident rays over the width of the fiber, we consider an azimuthal and a longitudinal distribution being generated by each ray we trace through the fiber. This approach is equivalent to having varying BCSDFs over h in Figure 4.4, which makes our model compatible with the far-field BCSDF framework, yet

enables near-field effects. In [152], the approach we use is classified as a near-field scattering model with constant incident illumination. Hence, we can leverage previous work, keeping the R, TT and TRT terms identical, but adding lobes for TrT, $TttT$, $TtrtT$, $TrRrT$ and $TttRttT$ paths, indicated in Figs. 4.4 and 4.12. Note that our framework can also handle general higher-order scatterings, but in our experiments, the contributions from those paths were insignificant.

It remains to find the forms of azimuthal and longitudinal scattering functions for each lobe. Conceptually, given the incident position and direction of a ray that enters the fiber, we ray-trace its chief specular ray [23, 154] on the 2D azimuthal and longitudinal cross sections of the fiber. We consider the direction at which the chief specular ray leaves the fiber as the center of a reflectance lobe, and accumulate the attenuation factors, and the azimuthal and the longitudinal roughnesses along the specular path, in the form of a Gaussian outgoing lobe. This is shown for the $TttT$ path $PQQ'P$ in the azimuthal section in Figure 4.12. In practice, rays are traced using closed-form analytical formulae. Longitudinal distributions are further simplified to conform to previous work, reducing to Gaussians with offsets/width for effective cuticle tilt/roughness. Finally, for those paths that enter the medulla, and are *scattered* by it, we must also include a scattered lobe (see broad yellow lobe at Q' in Figure 4.12). We precompute medulla scattering separately for 2D azimuthal/longitudinal profiles (Figure 4.13), based on [26], reducing scattering to a table lookup.

Formally, the near-field scattering distribution for each ray is,

$$\begin{aligned} S(\theta_i, \theta_r, \phi_i, \phi_r, h) =& \frac{\sum_p M_p^u(\theta_i, \theta_r) N_p^u(h, \phi)}{\cos^2 \theta_i} \\ &+ M^s(\theta_i, \theta_r, \phi) \frac{\sum_p N_p^s(h, \phi)}{\cos^2 \theta_i} \\ p \in \{R, TT, TRT, TrT, &TttT, TtrtT, TrRrT, TttRttT\}, \end{aligned} \tag{4.3}$$

where $\phi = \phi_r - \phi_i$, while M_p and N_p are respectively longitudinal and azimuthal scattering profiles that depend on the fur parameters in Table 4.1. We generalize equation 2.16 in a few respects. Note the h parameter (Figs. 4.4, 4.12) for near-field scattering. The superscripts s and u stand respectively for paths *scattered* by the medulla, and *unscattered*. (We will see that a single longitudinal M^s is adequate for all scattered paths p). The paths p include R, TT and TRT from previous hair models, as well as new terms specific to fur. Note that only unscattered lobes will be present for R, TT, TRT, TrT, $TrRrT$ paths that never transmit into the medulla, while $TttT$, $TtrtT$ and $TttRttT$ will have both scattered and unscattered lobes. Finally, we choose to replace the $1/\cos^2\theta_d$ term with $1/\cos^2\theta_i$ per [47] to mitigate the energy conservation issues at grazing angles at the cost of losing reciprocity, as analyzed by d'Eon et al. [.] This substitution also simplifies our rendering model, since the cosine terms are completely cancelled out in equation 2.14.

Simple Lobes: Consider rays that do not enter the medulla at all ($p \in \{R, TT, TRT, TrT, TrRrT\}$). In this case, there is no scattering from the medulla, and we can drop the superscripts. The

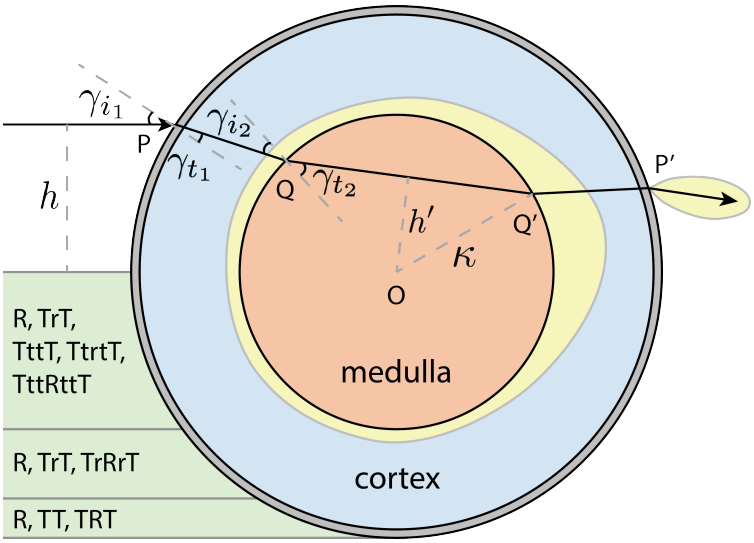


Figure 4.12: Illustration of evaluating the azimuthal scattering function for the type of path $TttT$.

azimuthal profile is simply

$$N_p(h, \phi) = A_p(h) \cdot D_p(h, \phi), \tag{4.4}$$

where A_p is the attenuation along path p, considering Fresnel terms on the cuticle and absorption along interior paths. D_p is the azimuthal distribution of scattered energy, a Gaussian with width determined by considering the roughness from all the surfaces the ray has interacted with. Both A_p and D_p share the same representation with unscattered lobes in equations 4.7 and 4.8. For the longitudinal profile of a ray, we follow [75, 23] and approximate it with a Gaussian distribution[5],

$$M_p(\theta_i, \theta_r) = G(\theta_r; -\theta_i + \alpha_p, \beta_p), \tag{4.5}$$

where α_p is the accumulative angular tilt of the chief specular ray on path p due to interaction with the cuticle scales, and β_p is the roughness for path p, which is empirically given by accumulating cuticle roughness. These expressions reduce to previous work for the lobes R, TT, TRT from the hair model. We now proceed to develop general formulae for paths that interact with the medulla.

Unscattered Lobes

We first discuss lobes that are not scattered by the medulla, giving expressions for M_p^u and N_p^u. Consider the azimuthal profile in Figure 4.12. Conceptually, we simply ray-trace the

[5]There are known energy leak issues at grazing angles since Gaussian lobes are unbounded, as analyzed and accurately solved in [23]. However, the accurate solution requires heavy computation and has numerical precision issues. Instead, we simply fold energy cutoffs outside [-90, 90] degrees back (e.g. 93 degrees back to 87 degrees), which works well in practice.

chief specular ray incident at P in the 2D azimuthal cross-section, accumulating attenuation and surface roughness along the path to give the intensity and width of the outoging Gaussian lobe centered at P'. In practice, ray-tracing can be replaced with simple analytic formulae.

An interesting observation is that certain types of paths only happen in specific zones over the offset h (see Figure 4.12). For example, depending on the size of the medulla, the traditional hair model paths T, TT and TRT are only possible for large h, and must enter the medulla otherwise. In practice, we solve geometrically for the boundaries of these zones, corresponding to $\gamma_{i_2} = 90°$ and $\gamma_{t_2} = 90°$, and only consider the relevant paths within each zone.

Azimuthal Scattering Profile: The azimuthal profile is given by equation 4.4, but additional handling is required for chief specular rays that hit and enter the medulla ($p \in \{TttT, TtrtT, TttRttT\}$). We split the contribution into two terms: the multiple scattered light for which we compute M^s and N^s in the next sub-section based on precomputed medulla scattering; and the unscattered light considered here, which is simply attenuated as it passes through the medulla. For instance, in a $TttT$ path as shown in Figure 4.12,

$$N_p^u(h,\phi) = A_{Tt}(h) \cdot A_{tt}(h) \cdot A_{tT}(h) \cdot D_p^u(h,\phi). \tag{4.6}$$

Here, $A_{tt}(h)$ is the attenuation factor for a chief specular ray transmitting through the medulla (QQ' in Figure 4.12).

With this background, a simple expression can be written for general higher-order paths, which can also be used for our $TtrtT$ and $TttRttT$ lobes. If we regard p as a string of length n, while p_i ($i = 1 \ldots n$) represents each vertex in p, we can write,

$$A_p(h) = \prod_{i=1}^{n} F(p_i) \cdot \prod_{i=2}^{n} \exp(-\sigma_t(p_{i-1}p_i) \cdot |p_{i-1}p_i|) \tag{4.7}$$

where F is the (extended) Fresnel term, $|p_{i-1}p_i|$ is the length of segment $p_{i-1}p_i$, and $\sigma_t(p_{i-1}p_i)$ equals either $\sigma_{c,a}$ of the cortex or $\sigma_{m,s}$ of the medulla, as the segment's extinction coefficient.

Similarly, at every intersection, the direction of p alters by angle $\Gamma(p_i)$. Thus, the outgoing azimuth could be computed by accumulating these deviation angles as $\Phi_p(h) = \pi + \sum_{i=1}^{n} \Gamma(p_i)$, where π accounts for the inversion of the incoming direction. For the distribution term D, similar to [23, 154], we accumulate the roughness i.e., β^2 at each intersection p_i along the path p. Since the outgoing distribution is a Gaussian lobe G centered at $\Phi_p(h)$, we can derive an analytic form,

$$D_p^u(h) = G\left(\Phi_p(h), \sqrt{\sum_{i=1}^{n} \beta^2(p_i)}\right) \tag{4.8}$$

where $\beta(p_i) = 0$ if the intersection p_i is not on the cuticle. Explicitly, the width is given by the number of upper-case (cortex) letters, and is β for $p \in \{R\}$, $\sqrt{2}\beta$ for $p \in \{TT, TrT, TttT, TtrtT\}$, and $\sqrt{3}\beta$ for $p \in \{TRT, TrRrT, TttRttT\}$.

Longitudinal Scattering Profile: The unscattered longitudinal profile is given by equation 4.5. All that remains is to determine the center α_p and width β_p of the unscattered

lobes. We follow previous work for R and TT lobes, setting $(\alpha_R, \beta_R) = (\alpha, \beta)$ for R, corresponding to the cuticle tilt and roughness, while $(\alpha_{TT}, \beta_{TT}) = (-\alpha/2, -\beta/2)$. For other lobes, we approximate

$$\begin{aligned} \alpha_p &= \alpha_{TT} - n_R\alpha \\ \beta_p &= \beta_{TT} + n_R\beta + (n \bmod 2)(\beta/2), \end{aligned} \tag{4.9}$$

where n is the length of p, and n_R is the number of Rs appearing in p. The general idea is that, every reflection on the cuticle decreases its tilt angle and increases roughness. The final $\beta/2$ compensates for lobes exiting backwards, since they are usually wider than forward lobes [21]. Thus, $\alpha_p = \alpha$ for $p \in R$, $\alpha_p = -\alpha/2$ for $p \in \{TT, TrT, TttT, TtrtT\}$ and $\alpha_p = -3\alpha/2$ for $p \in \{TRT, TrRrT, TttRttT\}$. Similarly, $\beta_p = \beta/2$ for $p \in \{TT, TttT\}$, $\beta_p = \beta$ for $p \in \{R, TrT, TtrtT\}$ and $\beta_p = 2\beta$ for $p \in \{TRT, TrRrT, TttRttT\}$. Note that, to our knowledge, longitudinal lobes' parameters were *empirically given* in most previous work; equation 4.9 provides a reasonable approximation to extend prior work to general paths that may enter the medulla.

Scattered Lobes

In this sub-section, we consider azimuthal specular paths p, deriving the scattering functions N_p^u and N_p^s. In both cases,

$$N_p(h, \phi) = A(p, h) \cdot D(p, h, \phi), \tag{4.10}$$

where A is the attenuation along p, and D is the distribution of energy for ϕ, considering surface roughness for unscattered lobes and scattering inside the medulla for scattered lobes.

Zone partition: Our first observation is that certain types of paths are only likely to happen in specific zones over the offset h (see Figure 4.12). We partition h into three types of zones and mark different types of paths in each partition. The boundaries of these zones could be geometrically solved at $\gamma_{i_2} = 90°$ and $\gamma_{t_2} = 90°$ respectively. So, given an incoming offset h, we first find possible types of paths, then calculate each type's scattering functions.

Unscattered lobes: Our insight is that, attenuation and roughness terms are accumulated along the specular path p. If we regard p as a string of length n, while p_i $(i = 1 \dots n)$ represents each letter in p, we could write the unscattered absorption term as

$$A^u(p, h) = \prod_{i=1}^{n} F(p_i) \cdot \prod_{i=2}^{n} \exp(-\sigma_t(p_{i-1}p_i) \cdot l(p_{i-1}p_i)) \tag{4.11}$$

where F is the (extended) Fresnel term, l is the length of segment $p_{i-1}p_i$, and $\sigma_t(p_{i-1}p_i)$ equals either $\sigma_{c,a}$ of the cortex or $\sigma_{m,s}$ of the medulla, as the extinction coefficient along each segment.

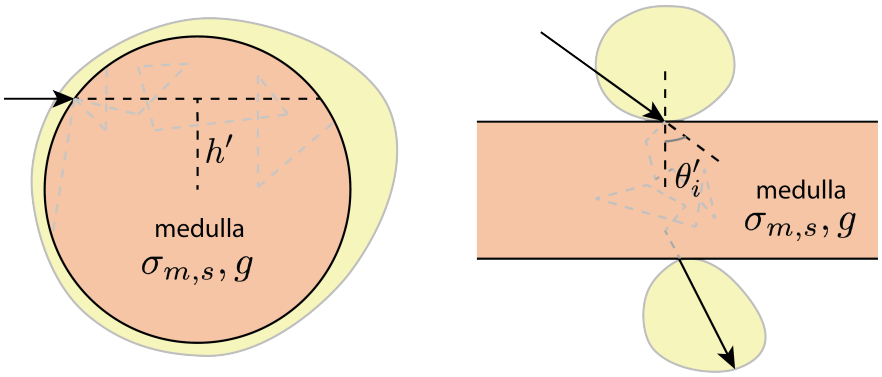


Figure 4.13: Precomputing medulla scattering. We enumerate $\sigma_{m,s}$ and g, and vary azimuthal offset h' and longitudinal incident angle θ_i' respectively.[6] We store the yellow-marked scattered lobe in every outgoing azimuth ϕ' and longitudinal outgoing angle θ_r'.

Similarly, at every intersection, the direction of p alters by angle $\Gamma(p_i)$. Thus, the outgoing azimuth could be computed by accumulating these deviation angles at each intersection as

$$\Phi(p,h) = \pi + \sum_{i=1}^{n} \Gamma(p_i), \tag{4.12}$$

where π accounts for the inversion of the incoming direction originally pointing outward.

For the distribution term D, similar to [23, 154], we accumulate the roughness i.e., β^2 at each intersection p_i along the path p. Since the outgoing distribution is a Gaussian lobe G centered at $\Phi(p,h)$, we can analytically write the distribution term D for unscattered lobes as

$$D^u(p,h) = G\left(\Phi(p,h), \sqrt{\sum_{i=1}^{n} \beta^2(p_i)}\right) \tag{4.13}$$

where $\beta(p_i) = 0$ if the intersection p_i is not on the cuticle.

Note that, when a path enters the bottom zone in Figure 4.12, the formulae for R, TT, TRT lobes are identical to previous work. For example, in our notation the $N_R(h,\phi) = F_1 \cdot G(-2\gamma_{i_1}, \beta)$ and $N_{TT}(h,\phi) = F_1^2 \exp(-\sigma_{c,a} \cdot \bar{PP'}) \cdot G(2\gamma_{t_1} - 2\gamma_{i_1} + \pi, \sqrt{2}\beta)$, which correspond to Eqn 7-8 in [75] and Eqn. 11-13 in [23]. While our TttT lobe has formula $N_{TttT}(h,\phi) = F_1^2 F_2^2 \exp(-\sigma_{c,a} \cdot 2\bar{PQ} - \sigma_{m,s} \cdot \bar{QQ'}) \cdot G(2\gamma_{t_1} - 2\gamma_{i_1} + 2\gamma_{t_2} - 2\gamma_{t_2} + \pi, \sqrt{2}\beta)$, which cannot be represented by any previous work. Here for simplicity we write Fresnel terms on the surface of cuticle and medulla as F_1 and F_2 respectively.

When the path $p \in \{TttT, TtrtT, TttRttT\}$ goes through the inner cylinder, scattering events happen, and this medulla scattering needs to be taken into account (the broad yellow lobe from Q' in Figure 4.12). We rely on precomputation, and make a number of significant

[6] In the precomputation, we assume all sub-paths are entering the medulla horizontally, since the azimuthal section is rotationally-invariant.

approximations to the full volumetric multiple scattering computation, to enable a practical rendering model.

Precomputation of Medulla Scattering: Our precomputation approach is similar in spirit to the empirical BSSRDF model by Donner et al. [.] However, since we use factored lobes, we precompute scattering profiles by 2D volumetric path-tracing separately for azimuthal and longitudinal components, for all possible combinations of scattering parameters, as illustrated in Figure 4.13. Here, we don't consider surface effects (reflection, refraction, Fresnel, etc) for the medulla; instead we compute these effects in the evaluation steps. In effect, we are precomputing 4D tables $C^N(\phi'; h', \sigma_{m,s}, g)$ azimuthally and $C^M(\theta'_r; \theta'_i, \sigma_{m,s}, g)$ longitudinally (1D profiles for 3D sets of parameters). We use primes to distinguish notation from the main parameters. Our precomputation is entirely scene-independent, and only needs to be done once. We also make C^N and C^M available online, so other researchers can use them directly. After compression, these tables takes up only about 20MB. The appendix discusses details.

Azimuthal Scattering Profile: The scattering lobe is usually large and diffusive, and therefore not significantly affected by the smaller effects of surface roughness. We also ignore refraction by the cortex-air interface, and assume that the light leaving the medulla is attenuated by a constant factor corresponding to the thickness of the cortex. Moreover, we assume that the precomputed scattered lobe from the medulla doesn't change its shape after transmitting outside. Finally, we assume that there is only one scattered lobe that is not further reflected by the cortex; the appendix discusses a first step towards relaxing this assumption.

In analogy to equation 4.6, the azimuthal scattering profile from a $TttT$ path is,

$$N_p^s(h, \phi) = A_{Tt}(h) \cdot A'_{tT}(h) \cdot D_p^s(h, \phi). \tag{4.14}$$

Note that the distribution D^s is not normalized, and accounts for the reduction in energy $1 - A_{tt}(h)$ due to the unscattered lobe already considered. The final attenuation A'_{tT} is now approximated simply as $\exp[-\sigma_{c,a} \cdot (1 - \kappa)]$, corresponding to the thickness of the cortex. We simply need to add one more attenuation term for $TtrtT$,

$$N_p^s(h, \phi) = A_{Tt}(h) \cdot A_{tr}(h) \cdot A'_{tT}(h) \cdot D_p^s(h, \phi). \tag{4.15}$$

Note that the scattered lobe arises only on the final rt segment; any scattering in the earlier segment will be considered as part of $TttT$. A similar expression can be used for the $TttRttT$ lobe. More generally, the attenuation of a scattered lobe consists of two parts. First, the energy reaches into the medulla. Second, the scattered energy is further absorbed, transmitting through the cortex.

The distribution term D for a scattered lobe is simply a query into the precomputed azimuthal scattering profile $C^N(\phi'; h', \sigma_{m,s}, g)$. Since we precompute the medulla as a unit circle, h' and $\sigma_{m,s}$ need to be normalized by radius κ, leading to

$$D_p^s(h, \phi) = C^N(\phi - \Phi_p(h); h'/\kappa, \sigma_{m,s}/\kappa, g), \tag{4.16}$$

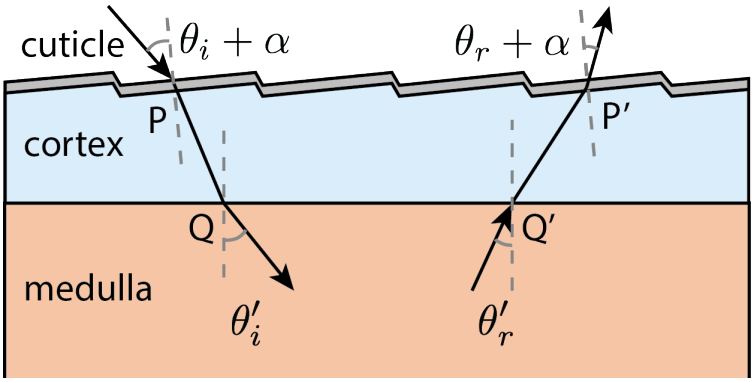


Figure 4.14: Illustration for computing longitudinal scattered lobe M^s. Refractions are considered here at P, Q, P' and Q'.

where Φ_p is the angle at which the ray enters the medulla.

Figure 4.12 shows a $TttT$ path on the azimuthal section. As illustrated, the unscattered lobe exits at P', carrying a Gaussian lobe due to surface roughness. The scattered lobe emerges from segment QQ', whose distribution is queried from the precomputed medulla scattering profile at the normalized offset $h'/\kappa = \sin\gamma_{t_2}$.

Longitudinal Scattering Profile: The longitudinal scattered lobe M^s is p-independent, because every type of path p has the same θ_i and θ_r. However, it is ϕ-dependent. As Figure 4.13 illustrates, the longitudinal profiles we precompute are for $\phi = 0$ (the upper lobe) and $\phi = \pm 180°$ (the lower lobe). We compute M^s at both azimuths, and linearly interpolate the results for any ϕ. Thus, for simplicity we omit ϕ in the following, and take $\phi = 0$ for illustration.

We query the precomputed longitudinal distribution for the medulla C^M, considering refractions through the cuticle and the cortex. As Figure 4.14 shows, θ_i' and θ_r' could be solved geometrically. Similar to the azimuthal case, we query the precomputed distribution at θ_r',

$$M^s(\theta_i, \theta_r) = \mu \cdot F_t \cdot C^M(\theta_r'; \theta_i', \sigma_{m,s}/\kappa, g), \tag{4.17}$$

where $F_t = (1 - F(\theta_r + \alpha, l)) \cdot (1 - F(\theta_r'))$ is the product of (extended) Fresnel transmittance, and μ is the normalization factor.

Implementation Details and Validation

Importance Sampling: Our importance sampling scheme is similar to [22], where we first perform a lobe selection, then sample this lobe azimuthally and longitudinally. Here, we treat all scattered lobes together. If an unscattered lobe p is chosen, we sample a Gaussian around its azimuthal outgoing center $\Phi_p(h)$ and its longitudinal outgoing center $-\theta_i + \alpha_p$. This leads to a near-perfect importance sampling scheme. If the (summed up) scattered lobe is chosen, we perform a cosine-weighted sampling longitudinally and a uniform sampling azimuthally,

	Wolf	Chipmunk	Cat	Fur pelt
# Strands	1.6 M	503 K	729 K	12.5 K
# Segs	8	8	10	5
Area light		✓		✓
Env. light	✓	✓	✓	
# Samples	1600	1600	2500	1024
Time	60.8 min	23.6 min	56.2 min	12.4 min

Table 4.3: Statistics for all of our scenes. We represent fur fibers using line segments for each fur fiber. # Strands is the number of fur fibers, and # Segs is the number of segments along each fur fiber. # Samples is the number of samples per pixel.

taking advantage of the fact that the scattered lobe is smooth. For multiple importance sampling, which queries the PDF at a given outgoing direction, we first perform a lobe selection similarly, then compute the corresponding PDF value at the outgoing direction, depending on whether the scattered or unscattered lobe is selected.

Non-separable lobes: In [21], the non-separable lobes representation was introduced to accurately capture light scattering through a hair fiber. In this representation, the center and width of longitudinal lobes further depends on the relative azimuth ϕ. Indeed, there are quality improvements in rendering results from the original paper. However, we find that the main difference is the shape of the R lobe, which is the only one that spans a large range over azimuthal angles. Thus, when applied to our double cylinder model, we simply represent our R lobe as non-separable, and leave other lobes using the traditional representations.

Validation: We compare 2D profiles generated using our rendering model with measured data and simulations in Figure 4.7. Our rendering model closely matches the measured reflectance profiles, and has comparable error (Table 4.2) as a full simulation, in some cases even being closer to the measurements. This is not surprising, since the physical double cylinder model is exactly the same. Minor discrepancies are due to approximations, such as factoring longitudinal and azimuthal scattering profiles, empirical longitudinal lobe centers and widths, and medulla scattering approximations.

4.7 Results

In this section, we show results generated using our rendering model, and visual comparisons to previous methods. All results use our rendering model as a shader within Mitsuba [51], run on an Intel 6-core 3.6 GHz i7 4960X CPU, hyperthreaded to 12 threads. Statistics of each scene such as number of strands, number of samples and timings are listed in Table 4.3. The times are total wall clock running time, including global illumination; the cost of evaluating our model is comparable to that of the Marschner model, which produces almost identical timings. For all the scenes, all parameters are directly derived from our database

$\kappa = 0.85$ $\sigma_{m,s} = 0.75$	$\kappa = 0.85$ $\sigma_{m,s} = 1.5$	$\kappa = 0.85$ $\sigma_{m,s} = 3.0$	$\kappa = 0.85$ $\sigma_{m,s} = 6.0$	$\kappa = 0.25$ $\sigma_{m,s} = 2.58$	$\kappa = 0.5$ $\sigma_{m,s} = 2.58$	$\kappa = 0.75$ $\sigma_{m,s} = 2.58$	$\kappa = 0.99$ $\sigma_{m,s} = 2.58$

Figure 4.15: Renderings of the Cat scene under environment lighting using our rendering model, with (left) increasing scattering coefficients $\sigma_{m,s}$ and (right) increasing medulla size. Note the differences between the first and fifth images, where the fifth image produces a strong TT term on the eyebrows and a clear secondary highlight on the forehead, which are characteristics from the Marschner model, because it is using a small medullary index or radius $\kappa = 0.25$.

in Table 4.2, except for the absorption term $\sigma_{c,a}$ which accounts for different colors.[7] We also account for the fact that the medulla does not absorb light, by multiplying $\sigma_{c,a}$ by the medullary index κ. For wolf and chipmunk, since we don't have corresponding fur samples, we refer to parameters for dog and mouse from our database instead.

In comparisons, we consider [23] as a correct implementation of the Marschner model, so that the dark appearance generated is due to specularity of the model itself, rather than energy conservation issues. For the Marschner model, we use its optimized fitting parameters from Fig. 4.10, rather than ad-hoc settings from industry such as enlarging azimuthal roughness. For the Kajiya-Kay model, we enable global illumination by normalizing equation 2.15, since the model itself is not energy conserving.

Wolf: Figure 4.1 shows our rendering result for a Wolf model with a side-by-side comparison with the Marschner model. The wolf model is placed on a turn-table to demonstrate consistency of our rendering model, by rotating it in our accompanying video. The environment lighting is manually blurred prior to rendering. Insets are provided to compare with the Marschner and Kajiya-Kay models. All the renderings are path traced (for global illumination between fibers and the environment) using 1600 samples per pixel (spp) at a resolution of 1920×1080. Our double cylinder model produces a diffusive and saturated appearance, while the Marschner model is highly specular and dark. Note that, our method actually produces brighter highlights with the layered cuticle model than Marschner, but these are less *visually* obvious due to low local contrast. The Kajiya-Kay model produces a hard and solid appearance even with global illumination. Intuitively, compared to our model, it is visually similar to a BRDF vs BSSRDF (BCSDF) comparison.

[7] For comparisons, we use colored textures to assign $\sigma_{c,a}$ in the Marschner model for each fiber rooted at texture coordinate (u, v) as $\sigma_{c,a} = -\log(T(u, v))/4$, where $T(u, v) \in [0, 1]$ is the texture color at (u, v), considering each color channel.

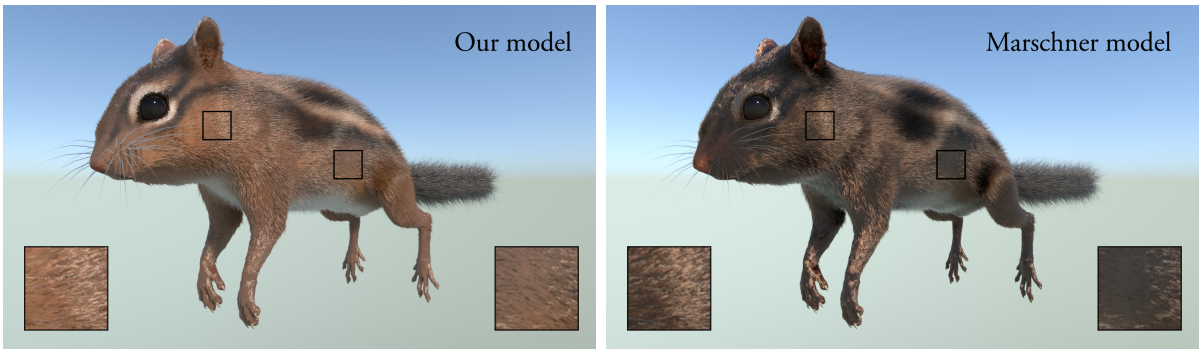


Figure 4.16: Renderings of the Chipmunk scene using (top) our double cylinder rendering model and (bottom) Marschner model illuminated with a strong area light and a dim environment light.

Cat: In Fig. 4.15, we render a close-up view of a cat head with depth of field effects under environment lighting, showing details of each fur fiber. We use our optimized fit parameters for cat fur in our database. All renderings use a sampling rate of 2500 spp with resolution 1024×1024. We can see that our rendering produces both a blurred area (top left of the eye) and distinct appearance (around the whisker) due to different arrangements and orientations.

We also show how an artist can manually vary key parameters of the model to get a range of appearances. Figure 4.15 shows renderings with varying scattering coefficients $\sigma_{m,s}$, as well as medulla size κ. We observe that, artistically, $\sigma_{m,s}$ closely controls color saturation, and κ determines the "specularity", or the extent of similarity between hair and fur. In the video, we rotate environment lighting to show color transitions of the cat fur from gold to dark orange under different lighting conditions.

Chipmunk: Figure 4.16 is rendered with a sharp and strong area light, and a relatively dim environment light. The skin of the chipmunk is dark colored. The Marschner model again produces unrealistically specular and dark appearance, since the light easily penetrates the fibers and hits the skin. However, primary (uncolored) and secondary (colored) highlights are still visible in the Marschner model. Both models are rendered using 1600 spp at resolution 1920×1080. In the video, we rotate the area light to see moving highlights, while other parts never get fully dark in our model.

Fur pelt: Figure 4.17 contains a pelt of fur placed on a checker board rendered using 1024 spp with resolution 1024×1024. A large area light illuminates the pelt from the top-left. Our model gives a realistic diffusive and saturated appearance, while the Kajiya-Kay model looks hard and solid. The Marschner model produces classic primary and secondary highlights, but leaves other regions black. By blending a diffuse lobe into the Marschner model as proposed by [153] (a solution that is widely adopted by the industry), one can generate a diffusive appearance. However, the blending technique lowers the intensity of the original lobes in the Marschner model, especially for the reflected lobe R, which leads to a flat appearance. Furthermore, this approach is empirical and the existence of the diffuse lobe cannot be explained physically.

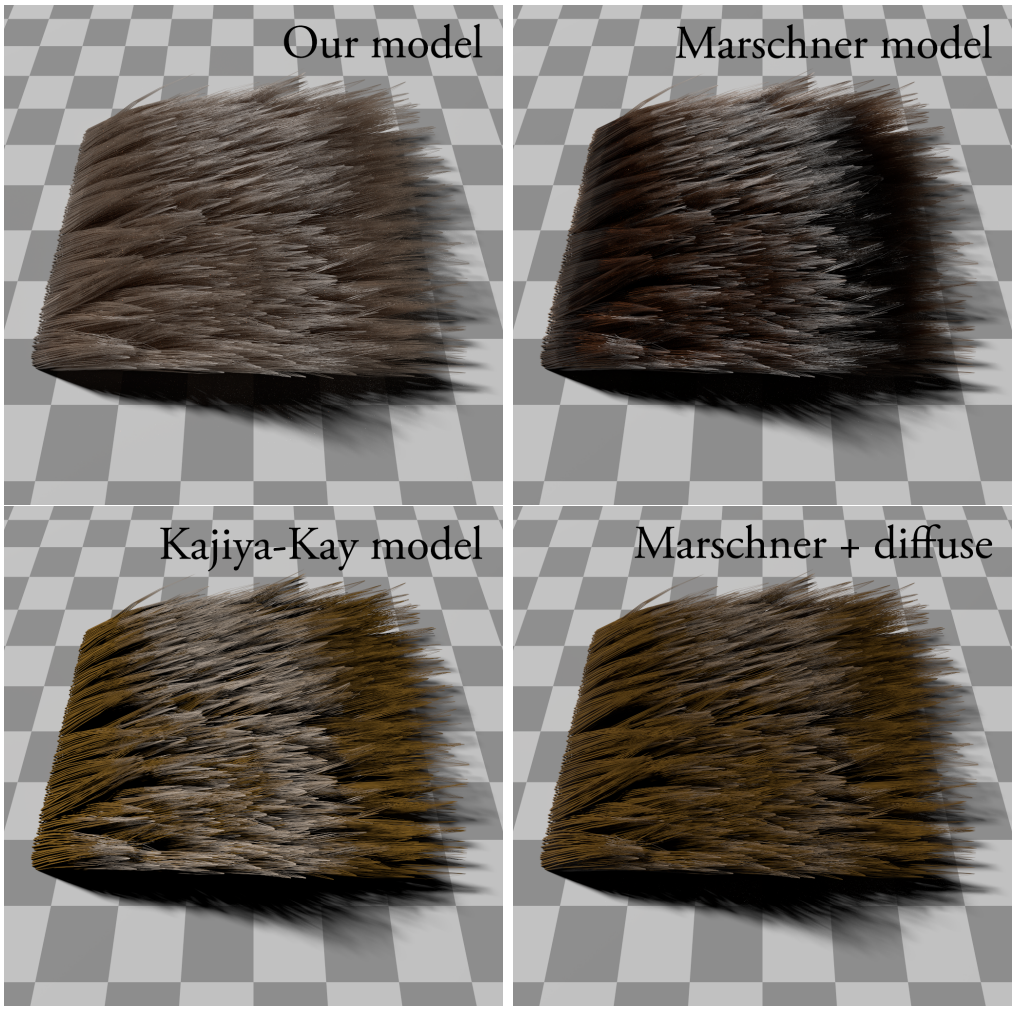


Figure 4.17: Renderings of the Fur pelt scene under area lighting. (Top left) Our rendering model. (Top right) Marschner model. (Bottom left) Kajiya-Kay model. (Bottom right) Marschner model blended with diffuse lobe.

4.8 Improvement: Simplified Near Field BCSDF Model

In previous sections, we have revealed the structural differences between hair and fur fibers, and have developed a double cylinder model correspondingly. However, fur rendering is still complicated due to the complex scattering paths through the medulla.

In this section, we focus on simplicity. We develop a number of optimizations that improve our double cylinder model's efficiency and generality without compromising accuracy, leading to a practical fur reflectance model. Similar to our previous approach, we validate our simplified model against measured data in Figure 4.24 and Table 4.7. Furthermore, we will

Parameter	Definition
η	refractive index of cortex and medulla
κ	medullary index (rel. radius length)
α	scale tilt for cuticle
β_m	longitudinal roughness of cuticle (stdev.)
β_n	azimuthal roughness of cuticle (stdev.)
$\sigma_{c,a}$	absorption coefficient in cortex
$\sigma_{m,s}$	scattering coefficient in medulla
$\sigma_{m,a}$	absorption coefficient in medulla
g	anisotropy factor of scattering in medulla
l	layers of cuticle

Table 4.4: Parameters used in our BCSDF model.

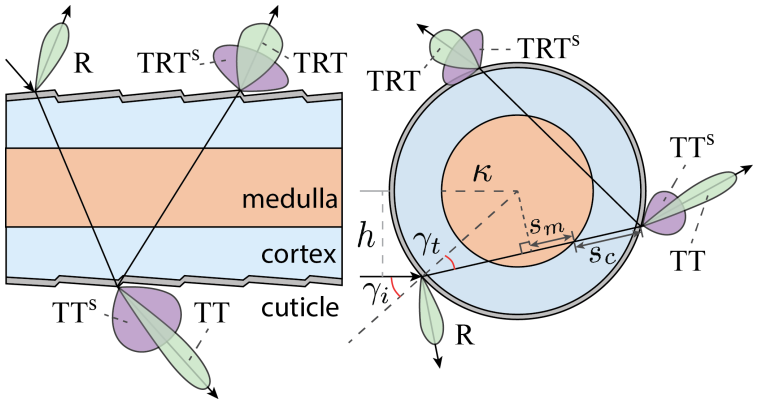


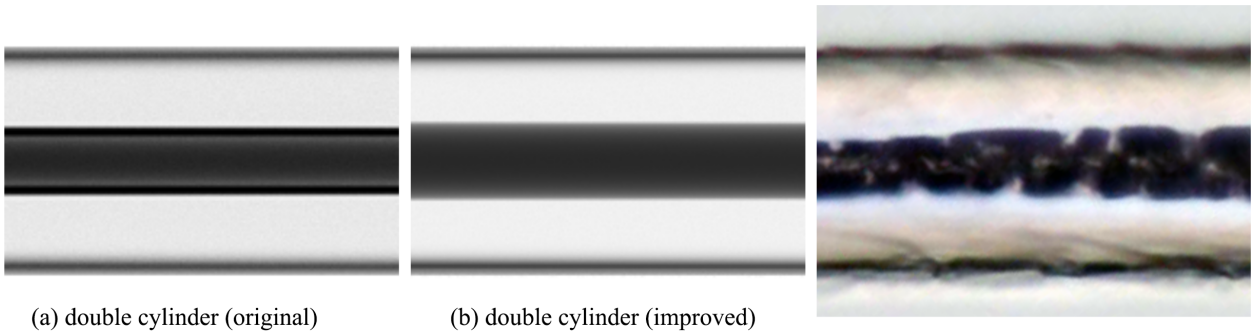
Figure 4.18: Our improved fur reflectance model with unified indices of refraction (IORs).

also describe an efficient piecewice analytic BCSDF model that unifies near and far field rendering in Section 4.9.

Overview

Our first observation is that, different indices of refraction (IORs) of the cortex and the medulla are mostly responsible for complex paths and lobes. However, in most of the fitting results presented earlier, the IORs between the cortex and the medulla are close. The similarity indicates that, complex types of paths such as TrT and $TtrtT$ are often too weak to be observed. Furthermore, Figure 4.19 shows a rare and extreme case where the IORs are clearly different. In this case, our original model with different IORs of the cortex and the medulla is supposed to be more accurate, but still fails to match the ground truth.

Based on this observation, we unify the IORs for the cortex and the medulla. Despite the assumption, this leads to a much simpler model, an analytic solution, and comparably accurate results as our previous model. With unified IORs, the light path no longer reflects



(a) double cylinder (original) $\eta_c = 1.45,\ \eta_m = 1.0$

(b) double cylinder (improved) $\eta = 1.45$

(c) microscopic photo

*Figure 4.19: Setting the medulla's index of refraction different from the cortex results in dark edges at the interface between them (a) for a fur fiber of polar bear lit from behind with directional lighting (*1.45 *for the cortex and* 1.0 *for medulla). This deviates from the photometric ground truth, even though polar bears' medullas are filled with air inside complex structures (c). Our local illumination model with unified IOR (*1.45*) does not have this problem (b). Microscopic photo courtesy of Carrlee et al. [8].*

or changes direction at the interface between the cortex and the medulla. Our model now shares the same light paths from hair models[8] — R, TT and TRT. The only difference is that, TT and TRT paths can be scattered passing through the medulla, forming scattered lobes TT^s and TRT^s. Thus, we write our near field BCSDF as:

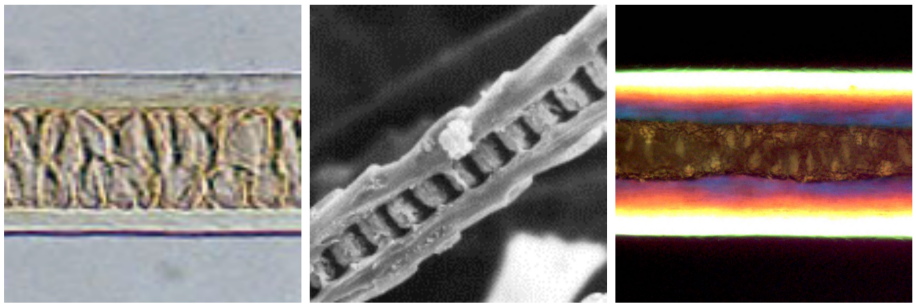
$$S(\theta_i, \theta_r, \phi_i, \phi_r, h) = \frac{S_R + S_{TT} + S_{TRT} + S^s_{TT} + S^s_{TRT}}{\cos^2 \theta_i}, \tag{4.18}$$

where $S_p = M_p(\theta_i, \theta_r)N_p(h, \phi)$ represent unscattered lobes and $S^s_p = M^s(\theta_i, \theta_r, \phi)N^s_p(h, \phi)$ represent scattered lobes. p is from the set of reduced types of paths $\{R = 0, TT = 1, TRT = 2\}$. S^s_R is always zero.

Apart from unifying the IORs, we improve our double cylinder model by introducing the medulla's absorption and different longitudinal/azimuthal roughness. Figure 4.20 demonstrates that the medulla can absorb light because of its complex internal structure, and that there are cases where pigments are found within the medulla [8]. We also take different longitudinal/azimuthal roughness into account, since these differences are often observed [34] and used in practice [23, 10].

Table 4.4 lists all the parameters used in our model, Figure 4.18 (c) illustrates all the lobes in our BCSDF model, and Figure 4.21 shows decomposed renderings using each lobe. Except for the R lobe that does not pass through the cortex, all other lobes produce colored appearance. The medulla blocks most TT light paths from previous hair models, thus producing a dark TT lobe and bright TT^s lobe. Though much simpler with only 5 lobes, our improved model is slightly more accurate, as validated in Table 4.7. Furthermore, individual

[8]We also need to consider attenuation by absorption in the medulla, in the formula for the R, TT and TRT lobes, as given in Table 4.6.



(a) Mounted medulla (b) Thick medulla (c) Pigmented medulla

Figure 4.20: The medulla absorbs light due to both its complex structure and the pigments inside. (a) Photograph of a fur fiber with the medulla filled with a mounting medium to minimize medulla's scattering. We can still see the structure of the medulla. (b) Solid ladder-like medulla which is comparably thick as the cortex. (c) Pigments are found filling brown bear's medulla, as reported by Carrlee et al. [8].

(a) All (b) R (c) TT (d) TRT (e) TT^s (f) TRT^s

Figure 4.21: (a) A rendering of a lock of hair with medulla. (b-f) With the unification of IORs, our reflectance model has only 5 lobes. From left to right, results rendered using lobe R, TT, TRT, TT^s and TRT^s, with path traced global illumination (Section 4.10).

lobes in our model are more efficient to evaluate, replacing more costly implicit ray tracing with closed-form expressions (Section 4.8, Figure 4.26). The pre-computed data for scattering lobes can be accurately compressed (Section 4.8) to make storage negligible (much less than a megabyte in all).

Lobe p	Shift α_p	Roughness β_p
R	α	β_m
TT	$-\alpha/2$	$\beta_m/2$
TRT	$-3\alpha/2$	$3\beta_m/2$
Lobe p	Distribution M^s	
TT^s	$\mathrm{lerp}_\phi(C^M_{\phi=0}, C^M_{\phi=\pi})$	
TRT^s	$\mathrm{lerp}_\phi(C^M_{\phi=0}, C^M_{\phi=\pi})$	

Table 4.5: Shift and roughness of longitudinal lobes.

Lobe p	Attenuation A_p or A^s_p	
R	F	
TT	$(1-F)^2 \exp\left(-\frac{2s_c\sigma_{c,a}+2s_m(\sigma_{m,a}+\sigma_{m,s})}{\cos\theta_d}\right)$	
TRT	$(1-F)^2 F \exp\left(-\frac{4s_c\sigma_{c,a}+4s_m(\sigma_{m,a}+\sigma_{m,s})}{\cos\theta_d}\right)$	
TT^s	$F \exp\left(-\frac{(s_c+1-\kappa)\sigma_{c,a}+\kappa\sigma_{m,a}}{\cos\theta_d}\right)$	
TRT^s	$(1-F)F \exp\left(-\frac{(3s_c+1-\kappa)\sigma_{c,a}+(2s_m+\kappa)\sigma_{m,a}+2s_m\sigma_{m,s}}{\cos\theta_d}\right)$	
Lobe p	Center Φ_p	Variance β_p^2 or Distribution D^s_p
R	β_n^2	
TT	$2(\gamma_t-\gamma_i)+\pi$	$2\beta_n^2$
TRT	$(\pi+2\gamma_t)+2(\gamma_t-\gamma_i)+\pi$	$3\beta_n^2$
TT^s	$\gamma_t-\gamma_i$	$C^N(\Phi^s_p-\phi)$
TRT^s	$3\gamma_t-\gamma_i+\pi$	$C^N(\Phi^s_p-\phi)$

Table 4.6: Attenuation and distribution of each azimuthal lobe.

Unscattered lobes (R, TT, TRT)

Since the light paths in our model no longer deviate from those in hair models, our model unifies hair and fur rendering (Figures 4.21 and 4.32). Thus, the unscattered lobes are very similar to the Marschner model.

Longitudinal unscattered lobes. As in previous work, we approximate the longitudinal scattering profiles for each lobe using an empirical Gaussian distribution $M_p(\theta_i, \theta_r) = G(\theta_r; -\theta_i + \alpha_p, \beta_p)$. Their centers and variances are listed in Table 4.5.

Similarly, the azimuthal unscattered lobes are evaluated using $N_p(h,\phi) = A_p(h)\cdot D_p(h,\phi)$ (Table 4.6). However, with the unification of IORs, we are able to derive closed-form expressions for both the attenuation term and the distribution term, rather than needing to use implicit ray tracing.

Azimuthal attenuation. As shown in Figure 4.18 (c), the R lobe ($p=0$) will be reflected by the cuticle directly, so it is attenuated by the Fresnel reflection $F(\eta', \gamma_i)$. The TT ($p=1$) and TRT ($p=2$) lobes both refract through the cuticle twice, thus attenuated

by two Fresnel transmissions, i.e. $(1-F)^2$. And the TRT lobe has an additional internal Fresnel reflection. Besides, both TT and TRT are attenuated traveling through the cortex and possibly the medulla, which are exponential falloffs with the distances $2s_c$ and $2s_m$ traveled within the cortex and the medulla, respectively. This is how colors are introduced. We list the attenuation terms for individual unscattered lobes R, TT and TRT in Table 4.6, and give a general representation[9] as

$$A_0(h) = F, \tag{4.19}$$
$$A_p(h) = (1-F)^2 F^{p-1} T_c T_m \qquad p \geq 1, \tag{4.20}$$

where $F = F(\eta', \gamma_i, l)$ is the Fresnel term with respect to cuticle layers l as in previous work, $T_c = \exp(-2ps_c\sigma_{c,a}/\cos\theta_d)$ and $T_m = \exp(-2ps_m(\sigma_{m,a}+\sigma_{m,s})/\cos\theta_d)$ are the attenuations within the cortex and medulla, respectively. The division by $\cos\theta_d$ is to account for elongated azimuthal paths when viewed from an oblique longitudinal angle. Here, different azimuthal offsets h decide different Fresnel terms F, as well as distances s_c and s_m a path travels, and thus the attenuation terms T_c and T_m.

Azimuthal distribution. For the distribution term D_p of unscattered lobes, since they are Gaussians $G(\Phi_p(h)-\phi;\beta_p)$ (Equation 4.8), what we need are their centers Φ_p and variance β_p. To find their centers, we follow the corresponding light paths, performing mirror reflections or refractions at intersections until the path leaves the double cylinder, as shown in Figure 4.18. In this way, the exiting azimuth Φ_p can be calculated, such that each refraction makes the path deviate its direction by $\gamma_t - \gamma_i$, and each internal reflection introduces a deviation of $\pi + 2\gamma_t$. For the distribution's variance, accumulating the squared roughness β_n^2 at each cuticle intersection is a simple multiplication with the number of intersections $p+1$. Similar to the attenuation terms, we also list the centers and variances of the distribution terms for lobes R, TT and TRT in Table 4.6, and give a generalized representation as

$$\Phi_p(h) = 2p\gamma_t - 2\gamma_i + p\pi, \tag{4.21}$$
$$\beta_p^2 = (p+1)\beta_n^2. \tag{4.22}$$

Scattered lobes (TT^s,TRT^s)

As introduced in Section 4.6, the scattered lobes involve two large pre-computed lookup tables C^M and C^N. In this subsection, we first describe how the pre-computed data can be compressed. Then we minimize interactions between the scattered lobes and the cuticle, so that querying the scattered lobe is simplified. Finally, we derive closed-form expressions for their attenuation and distribution terms (Table 4.6), so that the previous implicit ray tracing is no longer required.

[9]These general representations for all our attenuation and distribution computations also hold for arbitrary higher-ordered lobes, such as $TRRT$ and $TRRT^s$. However, we ignore them in our renderings for simplicity.

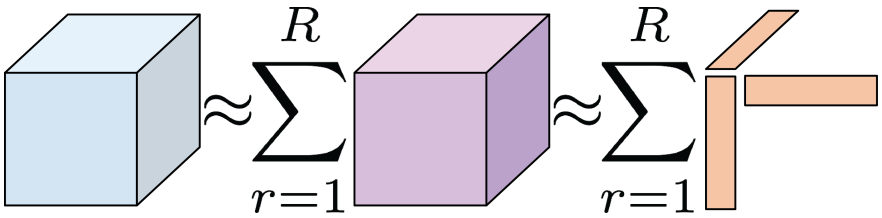


Figure 4.22: Illustration of tensor decomposition.

Compression. We treat these precomputed longitudinal and azimuthal scattering profiles C^M and C^N as 4D tensors, and refer to tensor decomposition techniques to compress them.

Tensors are high dimensional analogues to vectors or matrices, and can be regarded as multidimensional arrays. Tensor decomposition is a generalization of matrix singular value decomposition (SVD). In tensor decomposition, a d-dimensional tensor $\mathcal{A}$ is represented as a linear combination of R "simpler" tensors, each represented as the *tensor product* of d vectors:

$$\mathcal{A} = \sum_{r=1}^{R} \lambda_r \mathcal{A}^{(r)} = \sum_{r=1}^{R} \lambda_r \cdot \mathbf{a}^{(r,1)} \otimes \mathbf{a}^{(r,2)} \otimes \cdots \otimes \mathbf{a}^{(r,d)}, \tag{4.23}$$

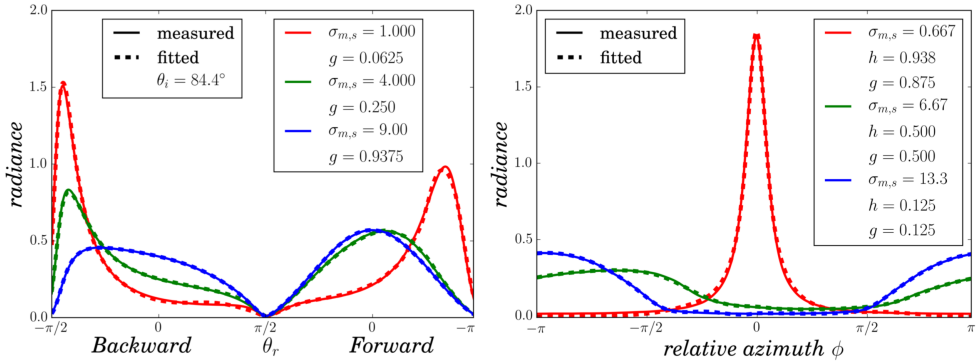
where the $(i_1, i_2, \ldots, i_d)$-th element of $\mathcal{A}^{(r)}$ is the product of the i_1-th element of $\mathbf{a}^{(r,1)}$, the i_2-th element of $\mathbf{a}^{(r,2)}$, ..., and the i_d-th element of $\mathbf{a}^{(r,d)}$ (Figure 4.22).

Here, R is the *rank* of tensor $\mathcal{A}$, and each $\mathcal{A}^{(r)}$ is rank 1. Similar to SVD for matrices, by assuming that the coefficients λ_r are sorted in decreasing order and keeping only the largest k coefficients, we are able to reconstruct tensor $\mathcal{A}$ approximately as

$$\mathcal{A} \approx \sum_{r=1}^{k} \lambda_r \cdot \mathbf{a}^{(r,1)} \otimes \mathbf{a}^{(r,2)} \otimes \cdots \otimes \mathbf{a}^{(r,d)}. \tag{4.24}$$

If a relatively small k is needed to appoximate tensor $\mathcal{A}$ accurately enough, we say that $\mathcal{A}$ is low rank. In this case, we only need to store $k \times d$ vectors to accurately reconstruct it, which is significantly less than the storage for $\mathcal{A}$ itself.

We apply this tensor decomposition scheme to compress the precomputed scattering profiles C^M and C^N, both with resolution $24 \times 16 \times 16 \times 720$, storing radiance at 24 values of $\sigma_{m,s}$, 16 values of g and 16 values of h or θ_i towards 720 outgoing directions. We use scikit-tensor, a Python module for multilinear algebra and tensor factorizations, to perform tensor decomposition using the alternating least squares (CP-ALS) algorithm. Our experiments show that, it usually takes less than one minute (single-threaded) to decompose either longitudinal or azimuthal precomputed data, with a maximum of 500 iterations. After the decomposition, we find that using up to rank 16 is good enough to accurately capture the complex shapes of all precomputed data.



(a) Longitudinal (b) Azimuthal

Figure 4.23: Comparison of measured (solid lines) and compressed (dotted lines) scattering profiles. Using our tensor decomposition scheme, the compressed profiles have good matches with the measured data over a wide range of parameters.

The resulting coefficients take only 150 KB in storage, which is negligible compared to the 600 MB raw data earlier. Figure 4.23 verifies the accuracy of our compression.

Note that previous tensor decomposition techniques for visual data [127, 134, 124] usually perform more complicated factorizations, using N-mode SVD with a core tensor. However, in our application, we find using a combination of rank 1 tensors suffices.

Longitudinal scattered lobes. The longitudinal scattered lobes are still interpolated between the precomputed lobes at $\phi = 0$ and $\phi = \pi$. However, we find that in Equation 4.17, the normalization factor μ is costly to compute but still approximate. Besides, the Fresnel transmittance cancels the normalization in most practical cases. Thus, we simplify the queries to use the incident and outgoing longitudinal angles directly as

$$M^s(\theta_i, \theta_r, \phi) = \text{lerp}_\phi(C^M_{\phi=0}(\theta_i, \theta_r), C^M_{\phi=\pi}(\theta_i, \theta_r)). \tag{4.25}$$

Azimuthal attenuation. Our azimuthal attenuation term A_p consists of two parts. The first part is the attenuation from the beginning to the point where the p-th segment starts intersecting the medulla, i.e. before the medulla scatters. Specifically, for TT^s ($p = 1$), we consider the first segment, and for TRT^s ($p = 2$), we consider the second rather than the first two, since the first segment's contribution is already accounted for in TT^s. In analogy to the unscattered lobes, we write the first part of the attenuation as $T_a = \exp(-[(2p - 1)s_c\sigma_{c,a} + 2(p - 1)s_m(\sigma_{m,a} + \sigma_{m,s})]/\cos\theta_d)$. The second part is the attenuation after the medulla's scattering, attenuated by the medulla of distance κ and by the cortex of distance $1-\kappa$. We assume no reflection/refraction events happen when exiting the cuticle. Thus, the second part of the attenuation becomes $T_b = \exp(-[\kappa\sigma_{m,a} + (1-\kappa)\sigma_{c,a}]/\cos\theta_d)$. The overall attenuation term is thus

$$A^s_p(h) = (1 - F)F^{p-1}T_aT_b. \tag{4.26}$$

Parameter	Unit	Bobcat	Cat	Deer	Dog	Mouse
κ	unitless	0.88	0.87	0.91	0.68	0.66
η	unitless	1.69	1.36	1.60	1.58	1.35
α	degree	5.48	3.65	3.52	2.94	0.55
β_m	degree	11.64	5.66	7.00	5.77	8.39
β_n	degree	7.49	1.34	4.53	18.94	2.80
$\sigma_{c,a}$	diameter^{-1}	0.64	0.06	1.39	0.01	0.04
$\sigma_{m,s}$	diameter^{-1}	1.69	2.47	2.51	2.44	1.34
$\sigma_{m,a}$	diameter^{-1}	0.17	0.12	0.09	0.00	0.06
g	unitless	0.44	0.60	0.46	0.26	0.36
l	unitless	0.47	0.44	0.45	0.60	2.36
NRMSE (original)		7.2%	5.3%	7.9%	9.1%	8.5%
NRMSE (improved)		6.8%	6.4%	7.1%	7.3%	4.7%

Parameter	Unit	Rabbit	Raccoon	Red fox	Springbok	Human
κ	unitless	0.79	0.65	0.86	0.82	0.36
η	unitless	1.47	1.19	1.49	1.48	1.20
α	degree	3.14	1.81	2.64	4.61	0.70
β_m	degree	11.91	7.44	9.45	8.02	2.05
β_n	degree	10.52	6.88	17.63	11.46	3.75
$\sigma_{c,a}$	diameter^{-1}	0.24	0.25	0.39	0.32	0.41
$\sigma_{m,s}$	diameter^{-1}	0.78	2.30	3.15	2.45	3.49
$\sigma_{m,a}$	diameter^{-1}	0.10	0.14	0.21	0.31	0.00
g	unitless	0.12	0.08	0.79	0.19	0.28
l	unitless	1.03	2.00	0.68	0.46	1.79
NRMSE (original)		8.4%	10.1%	6.3%	7.0%	19.3%
NRMSE (improved)		6.0%	9.7%	6.2%	8.1%	16.1%

Table 4.7: (Top) Optimized parameters fit from our measured data using our far field model. All length-related parameters are calculated assuming the azimuthal section of every fiber is a unit circle. All angle-related parameters are in degrees. (Bottom) Normalized RMS error of our original double cylinder model and our improved model.

Azimuthal distribution. For the distribution term $D_p^s(h,\phi) = C^N(\Phi_p^s(h) - \phi)$, we derive the direction Φ_p^s of the p-th segment, and use the difference angle $\Phi_p^s - \phi$ to query from the precomputed azimuthal scattering profile C^N. Similar to the unscattered lobes, the direction of the p-th segment that enters the inner cylinder is given by

$$\Phi_p^s(h) = (\gamma_t - \gamma_i) + (p-1)(\pi + 2\gamma_t). \tag{4.27}$$

Intuitively, Equation 4.27 is the result of one refraction into the outer cylinder for the TT^s lobe ($p = 1$), or one refraction plus one internal reflection for the TRT^s lobe ($p = 2$).

The final azimuthal scattered lobes are written as $N_p^s(h,\phi) = A_p^s(h) \cdot D_p^s(h,\phi)$.

In summary, for longitudinal scattered lobes, we simplify the interactions of the scattered lobes with the cuticle by ignoring the cuticle refraction. In this way, complex normalization is avoided along with Fresnel transmittance. For azimuthal scattered lobes, we assume that they originate from the center of the double cylinder and are attenuated evenly for all directions by the medulla and the cortex successively. So, compared to our previous model, our model does not have the additional diffusive lobe, and it accounts for absorption from the medulla. The simplicity of our model naturally leads to the ease of implementation. We provide implementation details in Section 4.10.

4.9 Improvement: Piecewise Analytic BCSDF Model

So far, we've derived a near field BCSDF. Now we show how to make a far field BCSDF approximation, which is especially efficient in reducing variance where hair or fur fibers are much thinner than a pixel. We then describe how to transition between near and far field models to make a multi-scale BCSDF that integrates per pixel. Our multi-scale BCSDF is the first model that is able to produce near field appearance when viewed close up, and requires no sampling when viewed far away.

Far field BCSDF model

To enable far field approximation, we need to integrate the near field azimuthal scattering profiles N_p and N_p^s over the azimuthal offset h. Both unscattered and scattered lobes share the same representation

$$N(\phi) = \frac{1}{2}\int_{-1}^{1} A(h) \cdot D(h,\phi)\,\mathrm{d}h, \tag{4.28}$$

For simplicity, we focus on the range $h \in [0,1]$ since it is always equivalent for symmetric queries (h,ϕ) and $(-h,-\phi)$. Thus, Equation 4.28 becomes

$$\begin{aligned} N(\phi) &= \frac{1}{2}\int_{0}^{1} A(h) \cdot D(h,\phi)\,\mathrm{d}h + \frac{1}{2}\int_{0}^{1} A(h) \cdot D(h,-\phi)\,\mathrm{d}h \\ &\triangleq N^{(+)}(\phi) + N^{(-)}(\phi). \end{aligned} \tag{4.29}$$

Unscattered lobes. We first look at the attenuation term. Our observation is that, the attenuation term A_p from Equation 4.20 can be treated as the product of four components: $(1-F)^2$, F^{p-1}, T_c and T_m. These four components are all functions of h — for the former two components with Fresnel terms, h defines different incident angles γ_i, and for the latter two absorption components, h decides the distances light travels within cortex and medulla s_c and s_m. Another observation is that, these components are either monotonic or smooth when h changes. So we start with partitioning h into a few segments (Figure 4.25 (a)). Then, for maximum accuracy, we linearize the entire first two components $(1-F)^2$ and F^{p-1} and

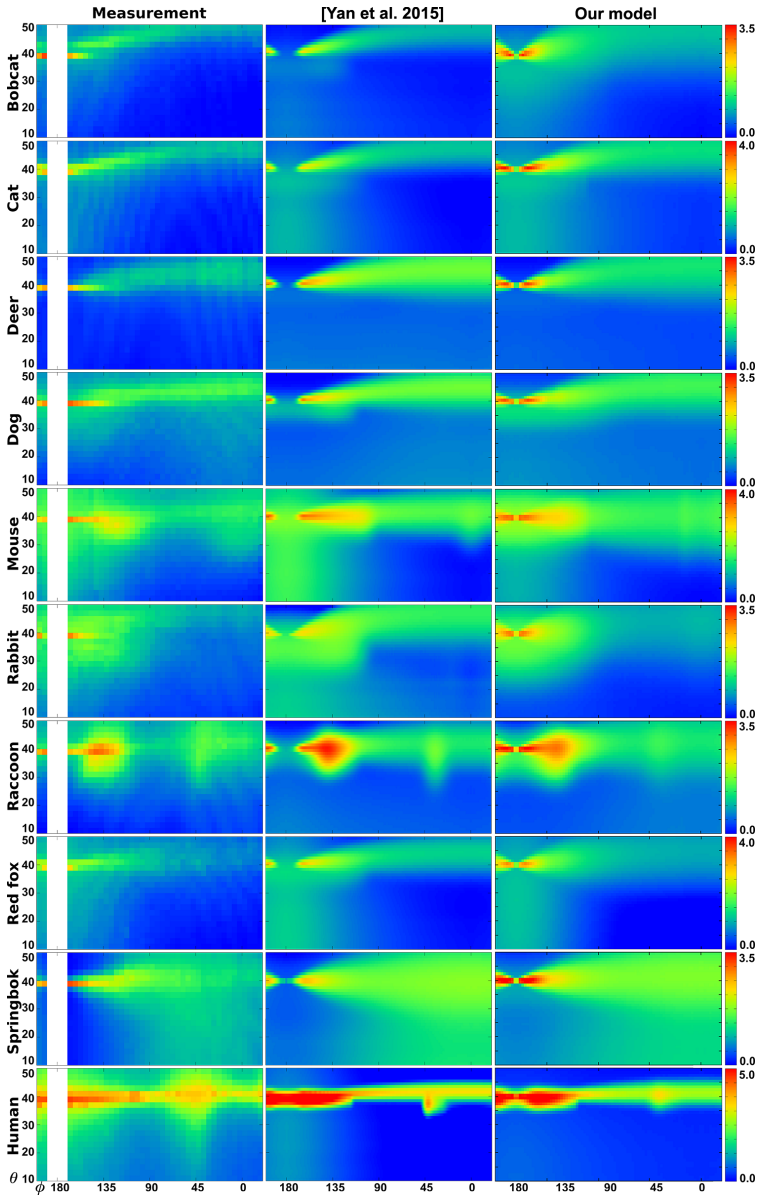


Figure 4.24: (Left) Reflectance profiles measured from different animals' fur fibers. (Middle) Synthesized profiles using the our original double cylinder model. (Right) Synthesized profiles using our analytic far field BCSDF model in Section 4.9. All profiles are scaled and displayed in logarithmic space for perceptual brightness.

the distances s_c and s_m in the latter two components. Thus, the attenuation term $A(h)$ becomes the product of two linear functions and two exponentials of linear functions.

Then we analyze the Gaussian distribution term $D_p(h, \phi) = G(\Phi_p(h) - \phi)$. Here we're not interested in its variance which is constant with h. Instead, we focus on its center that varies with h. As shown in Equation 4.21, when h changes, γ_t and γ_i change with it. So,

similar to the attenuation term, with segmented h, we are able to represent γ_t and γ_i with linear functions. Since a Gaussian is an exponential of squared variables, the distribution term ends up with an exponential of a quadratic polynomial.

With the piecewise polynomial representation inside both the attenuation term A_p and the distribution term D_p, we're able to represent the azimuthal scattering profile N_p for unscattered lobes in a simple form. Note that the product of two linear functions from A_p makes a quadratic polynomial, and the product of two exponentials of linear functions from A_p and the exponential of a quadratic polynomial from D_p together makes another exponential of a quadratic polynomial. So, we have the following form for unscattered lobes:

$$N_p^{(+|-)}(\phi) = \frac{1}{2}\sum_{i=1}^{n}\int_{h_{i-1}}^{h_i} Q_1(h)\cdot\exp(Q_2(h))\,\mathrm{d}h, \tag{4.30}$$

where Q_1 and Q_2 are quadratic polynomials. This can be easily solved analytically, as will be described with details in the Appendix.

Scattered lobes: The scattered lobes are similarly handled. For the attenuation term A_p^s in Equation 4.26, we linearize the Fresnel terms $1-F$, F^{p-1} as well as the distances s_c and s_m, resulting in the product of two linear functions and two exponentials of linear functions.

Since the distribution term D_p is queried rather than computed as a Gaussian, it is even simpler so that we directly linearize it. So the entire distribution term is a linear function.

Thus, the final result of the azimuthal scattering profile N_p^s for scattered lobes has the form

$$N_p^{s(+|-)}(\phi) = \frac{1}{2}\sum_{i=1}^{n-1}\int_{h_{i-1}}^{h_i} C(h)\cdot\exp(L(h))\,\mathrm{d}h, \tag{4.31}$$

where C is a cubic polynomial that is the product of linearized components $(1-F)$, F^{p-1} and D_p^s, and L is a linear function of the product from T_a and T_b. This integral can also be solved analytically with even simpler results than unscattered lobes.

Segmentation. We observed that when h is large, i.e. the incident position is away from the center, the linear terms change more rapidly. So we partition $h\in[0,1]$ quadratically, i.e. $h_i=\sqrt{i/n}$. For unscattered lobes, we find that using 5 segments is enough in most practical renderings, while using 8 segments generates indistinguishable scattering profiles. For scattered lobes, since they are even smoother, we find 4 segments good enough throughout all computations.

Acceleration. In practice, we usually don't have to integrate all n segments. Given a specific h, we immediately know its relative exiting azimuth $\Phi_p(h)$. For unscattered lobes, since the distribution term is a Gaussian around $\Phi_p(h)$, it is safe to assume that a path that is incident from h contributes only within this outgoing range $[\Phi_p(h)-3\beta_p, \Phi_p(h)+3\beta_p]$. Based on this observation, for a segment $h\in[h_1,h_2]$, we can limit its contribution within the range $[\min\{\Phi_p(h_1),\Phi_p(h_2)\}-3\beta_p, \max\{\Phi_p(h_1),\Phi_p(h_2)\}+3\beta_p]$. So we simply throw away all the queries with ϕ that are not within this range. In this way, each query for unscattered

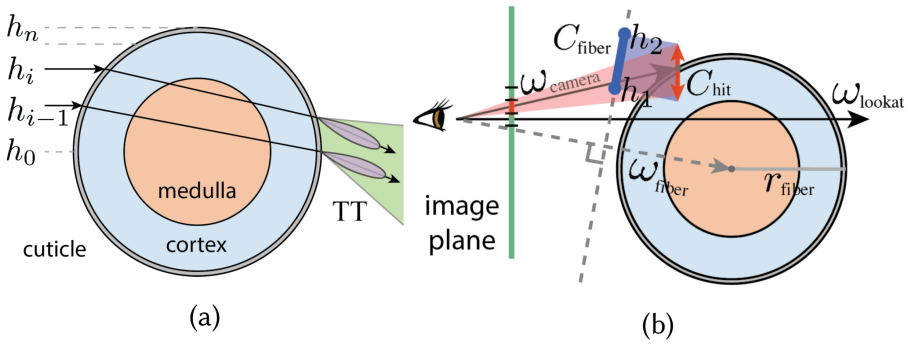


Figure 4.25: (a) Illustration of far field integration. The longitudinal section is partitioned into n segments. For each segment, we compute its contribution to the queried relative exiting azimuth. (b) Illustration of calculating a pixel's coverage for multi-scale rendering. A pixel (marked red on the image plane) is first projected to the hit point (red segment at the hit point), then projected again towards the fiber (blue segment).

lobes now requires an average of only $2-4$ integrations in practice. However, for scattered lobes, since the distribution term is precomputed for every direction, the acceleration scheme does not apply.

Validation. To verify the accuracy of all these simplifications / improvements, we re-fitted all the measured fur reflectance profiles and compared the NRMSE[10] with previous fitting results.

We use Ceres Solver [1], a nonlinear least squares minimizer, to fit the measured profiles. The fittings are performed in logarithemic space, with the cost function defined as the sum of all per-pixel differences of the log-measured and log-fitted values, divided by the range between the minimum and maximum log-measured values. The initial values of all parameters are manually set. The fitted profiles are generated using our far field model, with 8 segments for unscattered lobes and 5 segments for scattered lobes. Fitting each profile takes $2 \sim 3$ minutes in our test platform.

Figure 4.24 shows 3-way comparison of the measured data, our previously fitted profiles and new fitted profiles. From these profiles, we can see that even with only 5 lobes, our method is still able to produce similar forward scattered lobes (e.g. red fox) and backward scattered lobes (e.g. raccoon). Also, with the introduction of medulla absorption, and the unification of IORs thus reducing the complexity of light scattering, our model has much "cleaner" forward scattered regions near $(\theta = 10°, \phi = 180°)$ (e.g. mouse and rabbit), which cannot be handled previously. The introduced azimuthal roughness intuitively smoothes the fitted profiles azimuthally. It is especially obvious for the R and TT lobes, so that the high-intensity regions fit better (e.g. raccoon, rabbit and dog). Note that our new method may not be consistently better than previous over these regions. This is expected, since the

[10]Normalized RMS Error, or precisely, RMS error of the fitting result divided by the range of measured data.

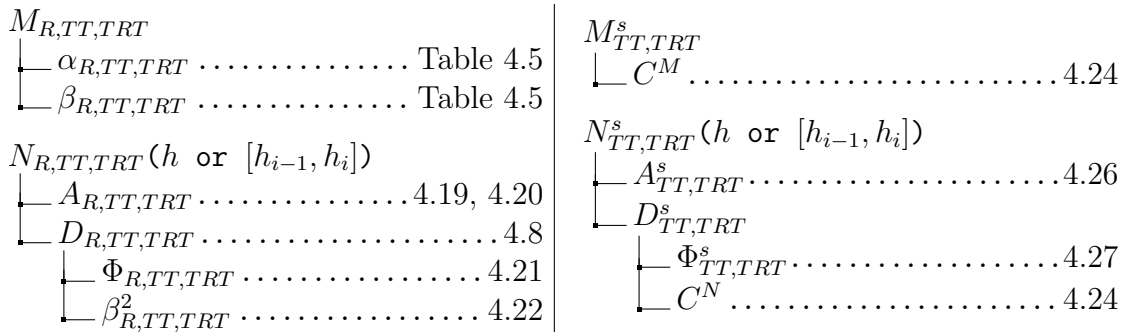


Figure 4.26: A dependency tree of variables for BCSDF evaluation. Equation numbers are marked. All the variables required for both unscattered and scattered lobes are also listed in Tables 4.5 and 4.6.

fitting procedure is aimed at global optimization.

One limitation of our method would be that, our results are slightly more blurred longitudinally, indicating larger longitudinal roughness fitted. This may relate to a cuticle scattering phenomenon observed as stripes in almost all measured profiles, and we leave it as future work. Also, there are cases (e.g. human) where neither our previous method or our improved method produce good fits, indicating that the double cylinder model can be further improved.

Table 4.7 lists all the fitted parameters and NRMSE values. As analyzed above, though our near field model makes several approximations for scattered lobes, and our far field model builds upon it with further piecewise linear approximations, our model still has better results in most cases. Note that our model is much simpler and more general, and the new parameters in our model (β_n and $\sigma_{c,a}$) provide better flexibility for artist control.

Multi-scale BCSDF model

Far field approximation is accurate when hair or fur fibers are thinner than a pixel. However, when viewed close up, a fiber's width may cover several pixels, i.e. each pixel actually covers a small range over the azimuthal section. In these cases, far field approximation will produce ribbon-like appearance (Chapter 2). To deal with these cases, we propose our multi-scale BCSDF model. We use each pixel's coverage on hair or fur fibers to decide the range of azimuthal offset h it covers, and integrate per pixel instead of per fiber. In this way, we integrate similar to far field approximation, but keep the accurate near field appearance.

Pixel-wise integration. Suppose we know that a pixel covers a range of azimuthal offset $h \in [h_1, h_2]$. We extend Equation 4.28 to integrate only within this range as

$$N(\phi) = \frac{1}{h_2 - h_1} \int_{h_1}^{h_2} A(h) \cdot D(h, \phi)\, \mathrm{d}h, \tag{4.32}$$

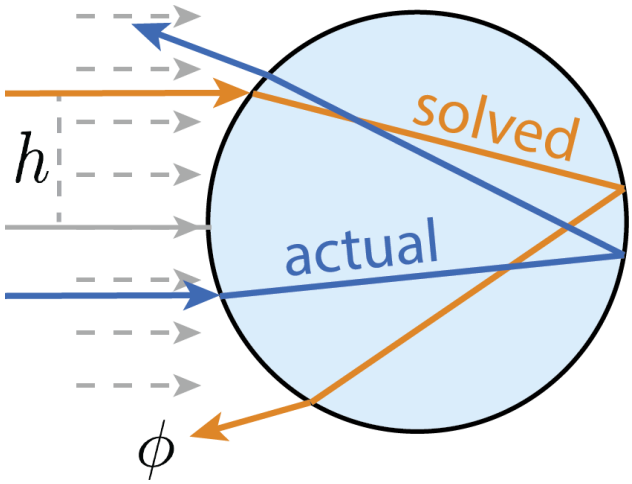


Figure 4.27: Validation of far field and multi-scale rendering. We render the same scene viewed close up (left three columns) and viewed from far (right three columns), using 1024 spp (top half) and 64 spp (bottom half). Timings are listed for 1024 spp and 64 spp, respectively. When zoomed out, our far field and multi-scale models perform around $1.5\times$ *slower than near field. When zoomed in, our multi-scale model performs closer to near field, since the range to integrate becomes smaller for each pixel.*

where the term $1/(h_2 - h_1)$ guarantees energy conservation. When a pixel fully covers the entire azimuthal section, Equation 4.32 degenerates to the far field case Equation 4.28. And in the limit case where h_1 and h_2 are infinitesimally close, it becomes the near field scattering representation $N(h, \phi) = A(h) \cdot D(h, \phi)$. These two cases indicate that our multi-scale BCSDF model bridges both near and far field scattering, and is consistent when scaling between them.

Calculating a pixel's coverage. Now that we have a multi-scale BCSDF model, what remains is to find a pixel's coverage $[h_1, h_2]$. Figure 4.25 illustrates the way to calculate it. Assuming that pixels are round rather than square on the image plane, we can tell how large a pixel's coverage is at the hit point in world coordinates using its diameter, denoted as C_{hit}. This is very similar to ray differentials [50]. Then, the projected pixel at the hit point becomes a disk, facing along the camera's look-at direction $\boldsymbol{\omega}_{\text{lookat}}$. We project the disk again towards the hit fiber, i.e. onto the direction $\boldsymbol{\omega}_{\text{fiber}}$. Finally, we compare it with the fiber's radius r_{fiber} in world coordinates to get the pixel's coverage. So, we have

$$C_{\text{fiber}} \approx (\boldsymbol{\omega}_{\text{fiber}} \cdot \boldsymbol{\omega}_{\text{lookat}}) C_{\text{hit}} / r_{\text{fiber}} \tag{4.33}$$

as the pixel's coverage in the azimuthal section of the fiber, with the same unit as $h \in [-1, 1]$. Here $\boldsymbol{\omega}_{\text{fiber}}$ is the direction from the camera to the fiber's center within the same azimuthal section with the hit point. The fiber's center can be calculated when performing ray-cylinder intersections. However, unless viewed from extremely close so that a fiber covers a very large area in the image plane, $\boldsymbol{\omega}_{\text{fiber}}$ is always close to the camera ray's direction $\boldsymbol{\omega}_{\text{camera}}$. So, in practice, we replace $\boldsymbol{\omega}_{\text{fiber}}$ with $\boldsymbol{\omega}_{\text{camera}}$ in Equation 4.33 for simplicity.

	Figure	#Strands	#Segs	#Samples	Time	Method
Hair Lock	4.21	1K	210	1024	3.7min	N/M/F
Raccoon	4.30	260K	22	1024	14.1min	M
Hamster	4.29	580K	15	1024	36.9min	N
Cat	4.31	267K	9	256	3.8min	M
Hair	4.32	53K	64	1024	17.3min	M

Table 4.8: Statistics for our scenes, all rendered in 720p, using different rendering methods (N for near field, M for multi-scale, F for far field). Each of the (# Strands) fur fibers is represented using (# Segs) line segments. # Samples is the number of samples per pixel.

After getting a pixel's coverage, the azimuthal range to integrate can be written as $[h - C_{\text{fiber}}/2, h + C_{\text{fiber}}/2]$. This range is then clamped to be within $[-1, 1]$, in case a pixel is much larger than a fiber's width, or it covers the boundary of a fiber. To integrate, we use the same segmentation scheme for far field approximation, clipping segments to be within this range.

Validation. We render the same insets using our near field, far field and multi-scale BCSDF models. As shown in Figure 4.27, when viewed far away, the differences between our three methods are barely visible, but the far field and multi-scale models produce significantly less noise. When viewed close up, our multi-scale model still generates the same results as compared to the near field model, but the far field results look flat. A similar effect is seen in Figure 4.30, where the multi-scale model produces the same appearance as the near field, but is much less noisy.

4.10 Results using the Improved Model

Implementation

In this section, we provide a brief summary of how to implement our improved near and far field model within global illumination renderers. We discuss two relevant aspects: evaluation and sampling.

BCSDF evaluation. Since our improved reflectance model unifies hair and fur rendering with only two additional scattered lobes, our BCSDF evaluation easily fits in a hair rendering system. We provide a dependency tree in Figure 4.26 of variables to compute, separating the classic R, TT and TRT lobes for hair and new scattered lobes TT^s, TRT^s.

When near field rendering is required, the azimuthal lobes are queried at specific offset h. On the other hand, for far field approximation or multi-scale rendering, queries happen on the ends of all segments. Note again that every step in the tree of Figure 4.26 is either analytic or queried, thanks to the unification of IORs.

Importance sampling. Similar to d'Eon et al. [22], our importance sampling works in four steps as follows:

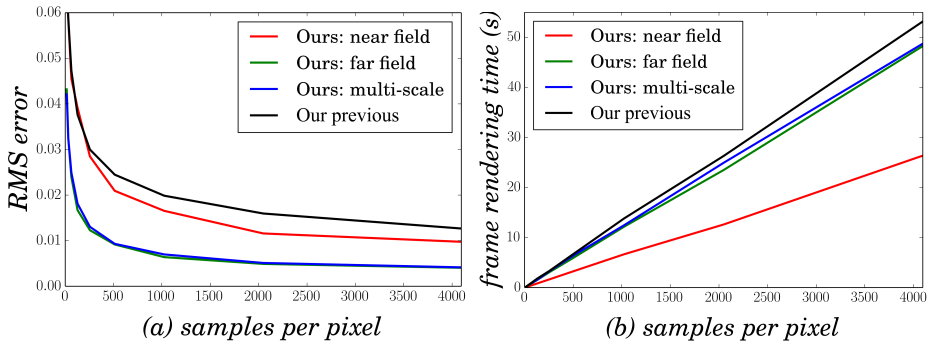


Figure 4.28: We compare our method with our previous method in (a) convergence and (b) frame rendering time. The experiments are conducted on the central 32×32 patch of the hair lock scene. Our near field model converges twice as fast as previous, and our multi-scale model converges an order of magnitude faster. Also, our near field model performs $2\times$ faster. Moreover, since our far field and multi-scale models require solving integrals rather than querying the integrand as near field models do, a slight performance drop is expected. However, they still outperform our previous model even though it is near field, indicating that our analytic integration is efficient. Also note that since the scene is viewed from far away, our far field and multi-scale models converge almost as fast in this case.

- Choosing azimuthal offset h. For near field scattering, since it is fixed, there's no need to choose. For far field approximation and multi-scale rendering, we randomly pick an h from the corresponding range, i.e. $[-1, 1]$ for far field and $[h_1, h_2]$ for multi-scale.

- Choosing a lobe p to sample. The lobes are weighted according to the energy they carry. Since the longitudinal integral of M_p is always 1, and the azimuthal integral of the distribution term D_p is also 1 for any selected h, the energy that lobe p carries depends on its attenuation term A_p. So, we calculate the attenuation term for all 5 lobes, and choose one with the probabilities in proportion to their values.

- Sampling azimuthally. Once the azimuthal offset h and the lobe p have been selected, for unscattered lobes, we immediately know how they distribute with center Φ_p and standard variance β_p. We perform a Gaussian sampling according to this distribution to get the relative exiting azimuth ϕ. Then the actual outgoing azimuth $\phi_r = \phi + \phi_i$ can be computed. For scattered lobes, since they're smooth, we sample them uniformly over all azimuthal angles.

- Sampling Longitudinally. This is very similar to the azimuthal case. Since we know which lobe is to be sampled, for unscattered lobes, we just need to sample according to the Gaussian M_p. For scattered lobes, we use cosine sampling.

- Calculating PDF and sampling weight. The final probability density function (PDF) is the product of PDFs sampling the selected lobe p azimuthally and longitudinally, followed by a conversion from (θ, ϕ)-measure to solid angle measure. The final sampling

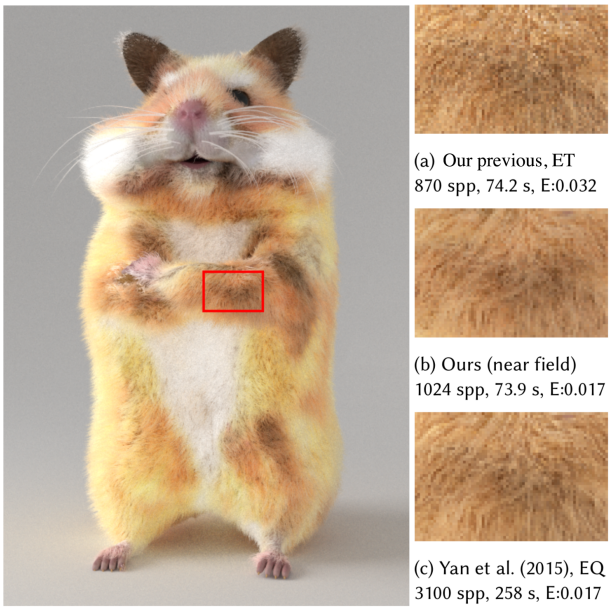


Figure 4.29: A Hamster *model rendered under studio lighting with a diffuse backdrop, using our near field model. Insets compare (b) our improved method with our previous method for (a) equal time (ET) and (c) equal quality (EQ) (E in the sub-captions stands for RMS error). The two models have different parameter spaces, thus small differences can be found in the EQ comparison. In the ET comparison, noise can be clearly seen when zoomed in. For equal quality, our method performs* 3.5× *faster.*

weight is the BCSDF value of the selected lobe p over the final PDF, then divided by the probability of selecting it. Since the unscattered lobes are importance sampled and the scattered lobes are usually smooth, the sampling weight is usually smaller than 2 in practice.

With the importance sampling scheme, we perform standard path tracing to determine accurate global illumination.

While our importance sampling method is similar to our previous method, one important different to notice is that our previous model has 11 lobes, most of which make a very small contribution to the final image. When a lobe with low energy is selected with low probability, the sampling weight will be large, and the result will be noisy (have high variance). In contrast, our reflectance model has only 5 lobes, each of which carries a significant amount of energy. Thus, even when comparing only our near field results, our method still has less noise (Figures 4.30, 4.28 and 4.29).

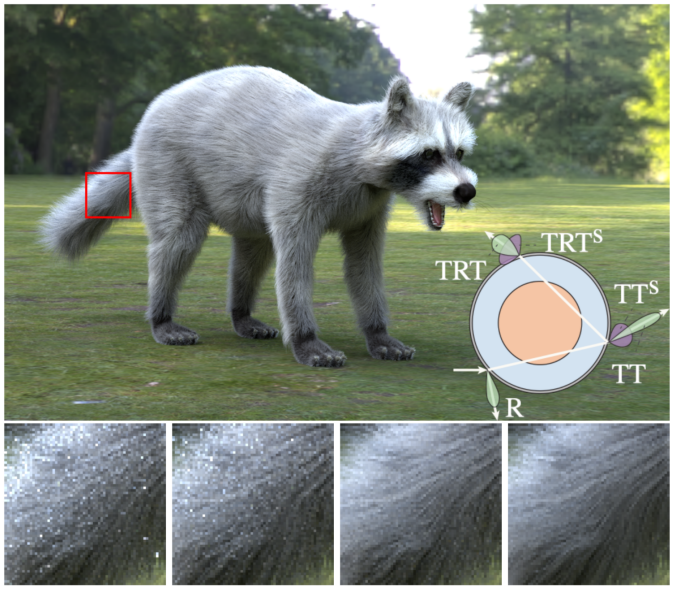


Figure 4.30: (Top row) A rendering of a raccoon model using our practical 5-lobe reflectance model (illustrated bottom right) and our multi-scale rendering scheme with 1K samples per pixel. (Bottom row) Insets rendered using different methods. We use 256 samples per pixel for equal sample (ES) comparison, and show equal quality (EQ) comparison with our previous model. Our multi-scale model performs more than 8× faster for equal quality while being practical and efficient. Even our near field model has significantly less noise than previous work.

Results

In this subsection, we show rendering results generated using our practical reflectance model, and compare them with previous work. We implement our model in the Mitsuba renderer [51]. Scene configurations, including number of hair or fur fibers and samples per pixel, are listed in Table 4.8. Parameters for raccoon, cat and human hair/fur models are taken from our best fit results in Table 4.7. The hamster model uses parameters from mouse. Since the measured data is grayscale only, to introduce color, we convert colored textures at different positions to absorption coefficients $\sigma_{c,a}$ in the cortex as before. All scenes are rendered using path tracing on an Amazon EC2 c4.8xlarge instance with 36 vCPUs. The source code and compressed pre-computed data are available on `http://viscomp.ucsd.edu/projects/fur2`.

We measure and compare the entire frame rendering time, including BVH traversal and ray-cylinder intersections. Even so, our near field reflectance model still performs around 3.5× faster than before in terms of equal quality comparison, and our multi-scale rendering scheme performs even better with up to a 8× speed-up. Note that since the parameter space in our reflectance model is different from our previous model, slight differences can be

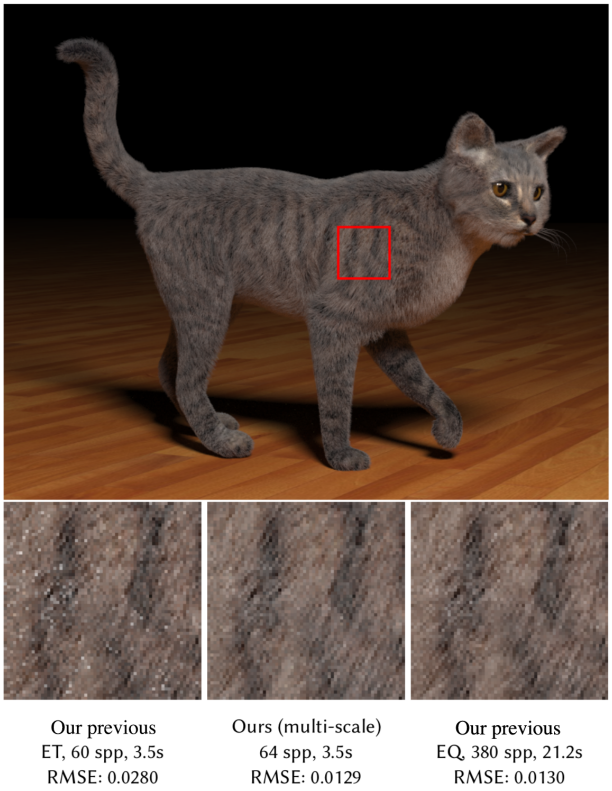


Figure 4.31: A Cat *model rendered multi-scale with an area light in front. We compare with our previous model for equal time and equal quality (same RMSE). For the same quality, our multi-scale BCSDF achieves a* 6.0× *speed-up.*

observed in their rendering results.[11]

Hair lock. Figure 4.21 shows decomposed renderings from each of the 5 lobes in our model, using the fitted parameters of human hair in Table 4.7. We can clearly see different lobes' contribution, indicating that our model is concise and effective. Figure 4.28 compares the convergence curves and rendering time using different methods. Previously, it is near field and does not require integration. However, our multi-scale model not only evaluates slightly faster, but also converges fastest among all the models. Our near field model evaluates fastest, but still converges 2× faster than before.

Raccoon. The raccoon scene is rendered with an HDR environment map. We use a ground projection scheme similar to Autodesk Fusion 360's implementation, so that the raccoon stands on an actual ground rather than floating. As shown in Figure 4.30, our near field model is capable of generating very similar diffusive appearance as compared to our previous model, but is much simpler and has less noise. Furthermore, our multi-scale rendering scheme converges significantly faster with minimum overhead. This is because the

[11]Hence, errors are computed with respect to the converged result for each reflectance model separately.

(a) Without medulla. ($\kappa = 0$) (b) With medulla. ($\kappa = 0.15$)

Figure 4.32: A hair model rendered with and without medulla using our multi-scale model. Our model unifies hair and fur rendering with the same light paths, regardless of the medulla's size. The difference between these two renderings is clearly visible, indicating the importance of the medulla as well as our scattered lobes TT^s and TRT^s, even with a small κ.

fur fibers are so thin that near field sampling is very inefficient, while multi-scale rendering integrates efficiently, successfully removing the high frequency noise.

Hamster. This scene (Figure 4.29) shows a hamster model, rendered under studio lighting with several area lights on top. The hamster model is located inside a capsule-like diffuse backdrop, encompassing the top, bottom and back sides. Since everything is diffuse, near field reflectance is efficient enough. We compare our near field model with our previous model which is also near field, showing that our model has better convergence because of its simplicity.

Cat. The cat model (Figure 4.31) is rendered using an area light in front. We compare our multi-scale model with our previous model. Our scene is rendered noise-free with only 256 samples per pixel.

Hair. Our reflectance model and multi-scale integration also work on human hair. As shown in Figure 4.32, even a small medulla ($\kappa = 0.15$) makes a difference in hair's overall appearance. Intuitively, this is because the light that goes through the medulla is spread more, making each hair more diffusive. The result indicates the importance of the medulla even for human hair, showing that the diffusive appearance comes not only from global illumination between hair fibers, but also within hair fibers. Note that the same light paths are computed through hair or fur fibers regardless of the medulla. Moreover, our multi-scale integration benefits both hair and fur rendering.

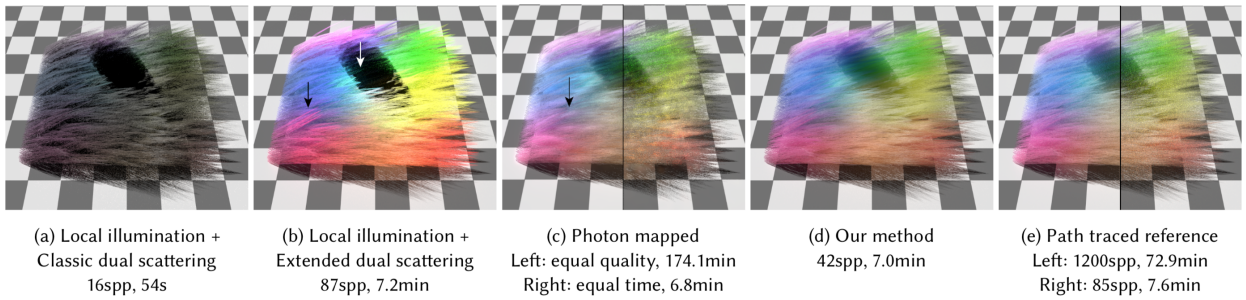


Figure 4.33: Rendering of the Pelt *scene using different methods, with a sphere casting a shadow onto it. The various colors are defined using a texture so that each fiber has a different color. (a) The classic dual scattering method fails to capture the scattered lobes from fur fibers, resulting in a dark appearance. (b) Our extended dual scattering handles scattered lobes, but is brighter and still produces hard and solid appearance, and does not have color bleeding effects. (c) Our method introduces a BSSRDF solution to the complex scattering, closely matching (d) the path traced reference, but is an order of magnitude faster.*

4.11 Efficient Fur Global Illumination Model

So far, we have completely introduced our double cylinder model. With our model, we are able to render animal fur accurately within a modern renderer, which simulates how light interacts with materials (local illumination) and bounces between objects (global illumination).

The global illumination part is responsible for the overall diffusive and saturated appearance of the hair or fur volume. In fact, compared to local illumination that considers how light interacts inside individual fibers, global illumination caused by light scattering within the fur volume is usually brighter and more visible, composing the main part of the appearance.

However, global illumination rendering is slow. It requires accurate simulation of hundreds of bounces of the light between individual fur fibers. Therefore, approximate methods are common, the most popular of which is the dual scattering technique [154], as introduced in Chapter 2. However, dual scattering does not work well for animal fur rendering. This motivates us to develop a novel BSSRDF (Bidirectional Surface Scattering Reflectance Distribution Function) solution to global illumination, addressing many of dual scattering's limitations. In this section, we will briefly analyze the failure cases of dual scattering to motivate our global illumination method. Then we provide a high level overview of our full global illumination model, specifying different components it consists of.

Failure of dual scattering

In dual scattering, there is an important assumption that the longitudinal lobes from hair BCSDFs (thus the averaged forward and backward lobes) are sharp. This is why it is

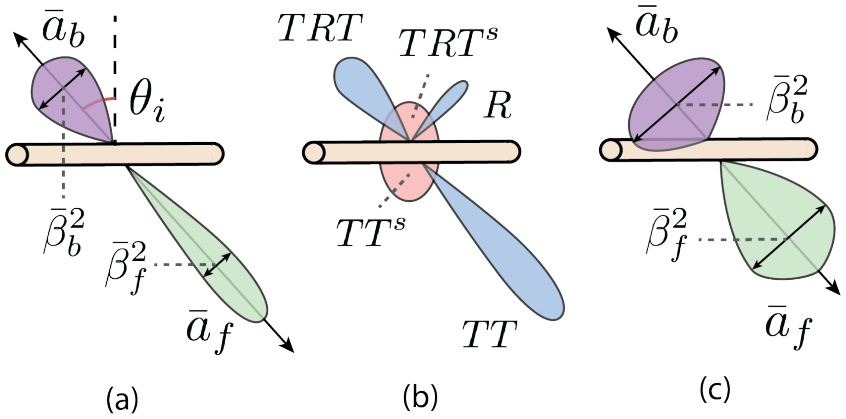


Figure 4.34: (a) Averaged forward/backward scattering internsities $\bar{a}_{f|b}$ and variances $\bar{\beta}^2_{f|b}$ for a single hair fiber. (b) All 5 longitudinal lobes in Yan et al. [141]. Note the scattered lobes TT^s and TRT^s that are too smooth and wide to follow the main path assumption. (c) Simple extension of dual scattering by calculating the contribution of scattered lobes to the forward/backward scattered lobes anyway.

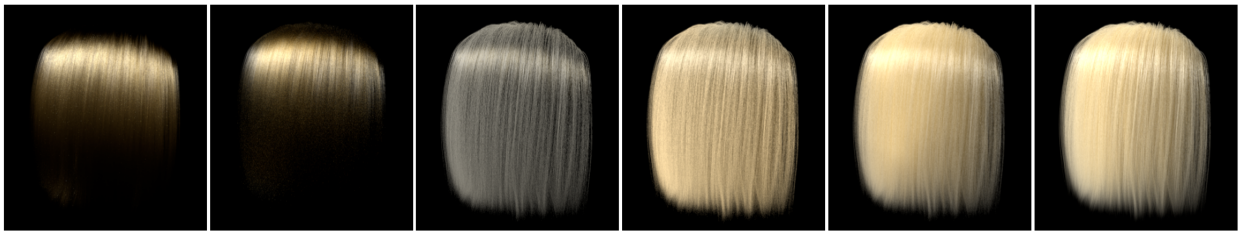


Figure 4.35: Classic dual scattering (a) works well on traditional hair models without medulla compared with path traced reference (b), but cannot capture scattered lobes from fur models with medulla (c). Our extended dual scattering (d) approximately handles the medulla and resolves the energy loss, but still doesn't match path traced reference (f) as well as our method (e).

reasonable that the light only transports along main paths. While generally true for human hair, when applied to animal fur, the model will break. The medullas within fur fibers produce smooth and diffusive scattered lobes, which do not exist in previous hair models and are not suitable to be abstracted using a forward and a backward lobe, as shown in Figure 4.34.

Even if we do assume light scattering only along main paths, the rendering results in Figures 4.33 and 4.46 indicate that dual scattering still cannot get us the correct light spread or color bleeding around the shading point $\mathbf{x}$. This is because dual scattering always adds approximate global illumination only to $\mathbf{x}$. Intuitively, the difference is analogous to BRDFs vs. BSSRDFs. This observation motivates us to model global illumination using BSSRDFs, as will be elaborated in Section 4.11.3 and 4.12.

Extending dual scattering

Since dual scattering only works for specular unscattered lobes, simply applying it to these lobes will result in energy loss from scattered lobes (see Figures 4.33 (a) and 4.35 (c)). To guarantee fair comparison with our approach later, we extend dual scattering to approximately handle the scattered lobes.

Our extension is to simply average the attenuation and spread of all lobes, regardless of whether they are spread or not (Figure 4.34 (c)). We calculate the averaged forward/backward attenuation $\bar{a}_{f|b}$ using Equation 2.18, but using all lobes including TT^s and TRT^s. We average the lobe-weighted average spread similar to Sadeghi et al. [**a**]s:

$$\bar{\beta}_{f|b} = \frac{\int_{\Omega_{f|b}} \sum_{p\in\{R,TT,TRT,TT^s,TRT^s\}} S_p \beta_p \, \mathrm{d}\boldsymbol{\omega}}{\int_{\Omega_{f|b}} \sum_{p\in\{R,TT,TRT,TT^s,TRT^s\}} S_p \, \mathrm{d}\boldsymbol{\omega}}, \tag{4.34}$$

assuming the stardard variances of the TT^s and TRT^s lobes as $\beta_{TT^s} = \beta_{TRT^s} = \pi/4$, since they are approximately uniformly distributed in the longitudinal section. The spread is pre-computed by uniformly sampling θ and ϕ and numerically calculating the integrals.

Note that, our extension to dual scattering is a bold assumption. It will result in unreasonably wide forward and backward lobes, but at least allows for energy conservation (See Figure 4.34). Moreover, it still makes the main path assumption, thus resulting in no color bleeding effects, as pointed out in Figure 4.33 (b). Furthermore, the brightness of the extended dual scattering is not stable (with fixed $d_f = d_b = 0.7$). Sometimes it will generate much brighter appearance than the reference (Figure 4.33) while sometimes being darker (Figure 4.44).

We compare the classic and our extended dual scattering in Figure 4.35 (a)-(d). We first use the classic dual scattering on human hair without medulla (a) together with local illumination. Since no scattered lobes exist, this is a case where the classic dual scattering works well, compared to the reference (b). Then we apply the classic dual scattering and our extended dual scattering with a medulla of width $\kappa = 0.36$ (c)-(d), also with local illumination, and compare them with our method (e) and reference (f). We find that the classic dual scattering suffers from severe energy loss. The extended dual scattering alleviates this issue, but generates a flat appearance. In contrast, our BSSRDF model, introduced next, generates a close match with the reference, as will be introduced next.

Model overview

To break the main path constraint, we represent our global illumination model as a combination of a BCSDF and a BSSRDF. The BCSDF also contains two components: direct illumination and dual scattering for multiple scattering from only specular lobes. The BSSRDF is responsible for all other multiple scattering events.

The direct illumination component describes light interaction with single hair or fur fibers. It is thus the hair or fur's BCSDF S_c in Equation 2.13. We name this part as the *local illumination component*.

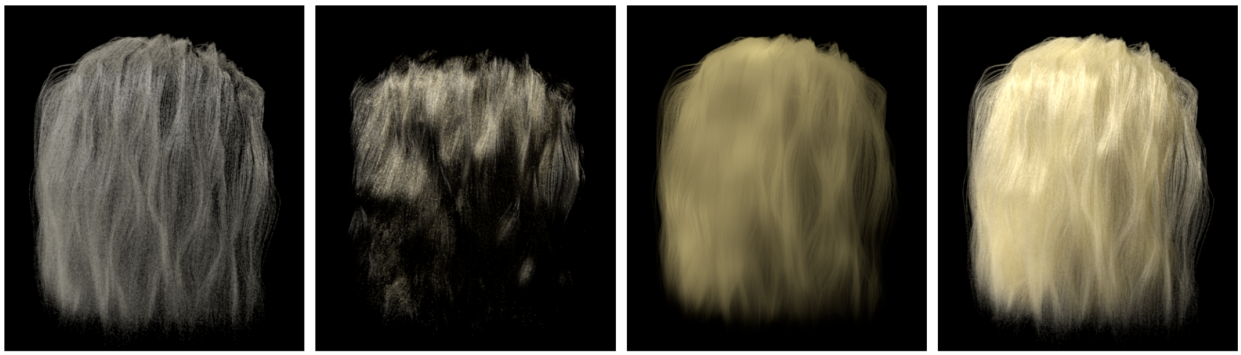


Figure 4.36: Decomposed components in our global illumination model. (a) Local illumination component S_c, including all 5 lobes: R, TT, TRT, TT^s and TRT^s. (b) Unscattered component S_{ds} captured by classic dual scattering, including R, TT and TRT lobes. (c) Scattered component S_{ss} represented using BSSRDF, including all light interactions that are related to TT^s and TRT^s lobes. (d) Our full model with all components.

Another part is the dual scattering of specular lobes S_{ds}. This is part of the global illumination. Since there are no scattered lobes, the main path assumption still holds. Specifically, we convert only R, TT and TRT lobes into forward and backward lobes using Equation 2.18, and use them to compute dual scattering's contribution. We define this part as the *unscattered component*. This part is slightly more accurate than the classic dual scattering method, since we also take the R lobe into account.

For the rest of the energy, at least one scattered lobe contributes. Since the scattered lobes are usually smooth and more isotropic than unscattered specular lobes, significant scattering happens. Thus, we use subsurface scattering to capture the scattering effects. Specifically, given the parameters of hair or fur, we convert them to subsurface parameters, then use the dipole method for rendering. We name this part as the *scattered component*. Note that, for hair which doesn't have a medulla, this component becomes zero. In this case, our method reduces to classic dual scattering.

Our final model can be represented as:

$$S_{\text{BCSDF}} = S_c(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}) + S_{ds}(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}), \tag{4.35}$$

$$S_{\text{BSSRDF}} = S_{ss}(x_i, \boldsymbol{\omega_i}, x_o, \boldsymbol{\omega_r}), \tag{4.36}$$

where S_c is individual hair or fur fiber's BCSDF. S_{ds} is the dual scattering approximated unscattered component, which is essentially another BCSDF. S_{ss} is our estimated BSSRDF scattered component[12]. For an intuitive illustration of different components, we visualize the decomposed appearance of each component in Figure 4.36.

[12]The BCSDF and BSSRDF components are defined in different domains, and they are integrated using Eqns. 2.13 and 2.3, respectively.

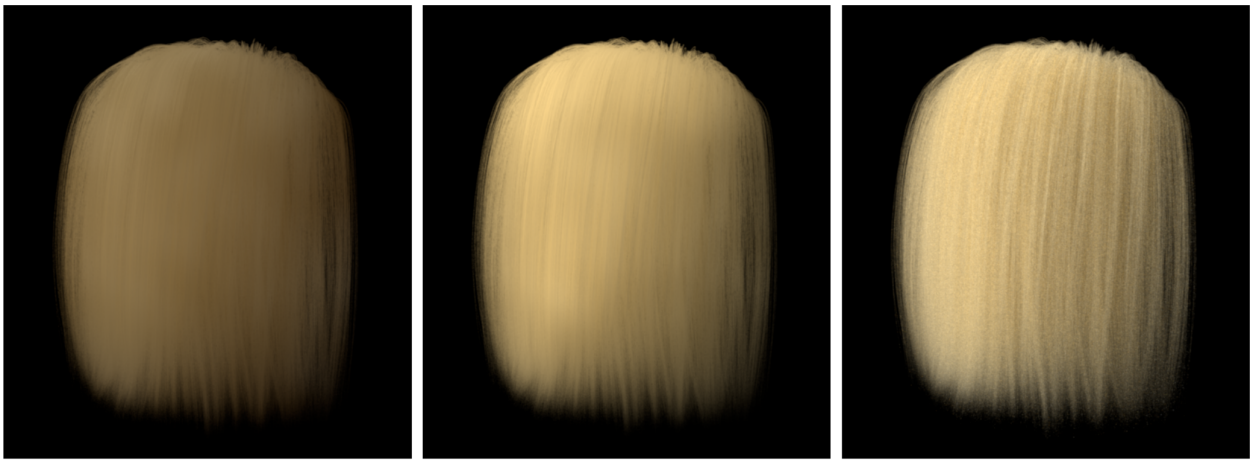


Figure 4.37: Rendering of scattered component using different methods. Our empirical BSSRDF model fits the path traced reference much closer, compared to the physically derived microflake model.

Though the idea of using a BSSRDF model to represent the scattered component may sound straightforward, it is far from easy, both in theory and in practice. First, BSSRDFs work only with actual surfaces. However, when it comes to hair or fur fibers, there is no concept of a surface. For this reason, distributing dipole samples for hair and fur and calculating associated areas with them is difficult. Second, there is no existing theory to convert physically based hair and fur properties into parameters of BSSRDFs. Third, the dipole solution to BSSRDFs only transports light locally in the shading point's neighbourhood. So, indirect lighting from the same hair or fur volume cannot be accounted for, resulting in dark areas with energy loss. In Section 4.12 and 4.13, we elaborate how to deal with all these difficulties.

Physically-based derivation of BSSRDF parameters

Theoretically, there is no existing BSSRDF model that is suitable to represent hair and fur volumes, since the hair and fur fibers define rather anisotropic scattering behavior. The closest match is the microflake model [54, 46], where the volume is assumed to be filled with randomly oriented flakes according to some distribution. Deriving a physically based model based on the microflake theory is not impossible. However, we find it does not work well in practice, and we turn to a data-driven approach to solve the problem, as will be introduced in the next section. Here we assume that the microflake model is the natural first step to try, so we describe it below and compare it with our BSSRDF model in Figure 4.37.

We think of each hair/fur fiber as a series of small "flakes" with their BCSDFs as phase functions [54]. However, this simple conversion is not well-defined in practice, mainly because of the high complexity of hair/fur fibers. The BCSDF of a hair/fur fiber consists of 5 lobes, R, TT, TRT, TT^s and TRT^s, which all have rather complicated shape and large variation with

the incoming light direction. However, in microflake theory, each microflake is completely opaque and mirror reflective. Nevertheless, it is still possible for us to derive microflake parameters by assuming local similarity, as discussed in the supplementary material.

We apply the derived parameters to render an actual scene, as shown in Figure 4.37. As expected, the result is still far from the path traced reference. The different shading distribution demonstrates the inability of individual flakes to represent hair/fur fiber segments, as analyzed above. Moreover, we also find a color difference as compared to the reference. This is because the color mainly comes from the absorption coefficient σ_a, which is usually two orders of magnitude smaller than the scattering coefficient σ_s. So, a slight inaccuracy in σ_a will result in significant color difference.

4.12 BSSRDF Approximation for Scattered Components

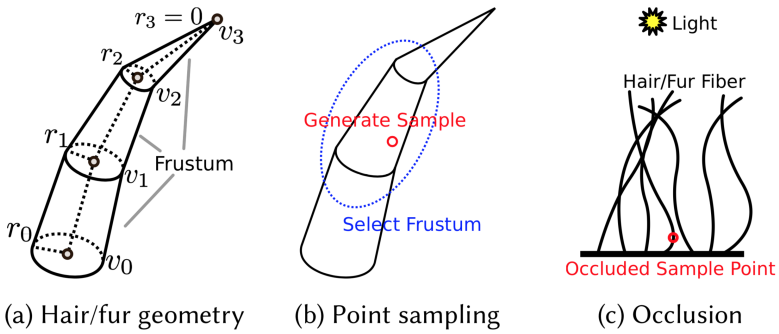
As discussed in Sec. 4.11, since deriving a practical physically based model to estimate the dipole parameters from the hair/fur parameters is challenging, we turn to a data driven approach. We propose a neural network structure to solve the parameter conversion problem. We chose a neural network over other methods because the dimensionality of the input and output spaces of the problem is high and the conversion function is likely to be non-linear, making simple fitting methods impractical.

In this section, we first generalize the dipole model for hair/fur geometry in Section 4.12 to be able to use the model in the first place. Then, in Section 4.12, we describe our neural network architecture for the parameter conversion problem. In Section 4.12, we describe how to train the neural network efficiently.

Generalizing BSSRDF for Fur/Hair Geometry

We extend the 2-pass algorithm introduced in Chapter 2 to enable hair/fur models in dipole rendering. The only missing component here is how to perform point sampling on a hair/fur geometry.

Figure 4.38(a) shows a hair/fur geometry representation, where each hair/fur fiber is represented by an array of vertices $v_0, ..., v_3$ with associated radiuses $r_0, ..., r_3$. The geometry represented by each 2 consecutive vertices is a frustum. To generate a sample point, we first randomly select a frustum from all frustums in the hair/fur geometry with a probability proportional to its side surface area. Then, a sample point can be obtained by a uniform sampling on the selected frustum's side surface as in Figure 4.38(b). Since the sample points generated this way may be highly occluded from the environment as illustrated in Figure 4.38(c), we remove those points that have zero estimated irradiance in the first pass in the octree construction process for efficiency.



(a) Hair/fur geometry (b) Point sampling (c) Occlusion

Figure 4.38: Sample placement for hair/fur geometry.

Neural Network for Parameter Conversion

In this section, we now describe our neural network architecture to estimate the parameters of a dipole model given the parameters of a fur/hair model. The appearance of a fur/hair model is decided by the 9 material parameters listed in Table 4.1. From our experiments, we find that only 5 parameters, κ, $\sigma_{c,a}$, $\sigma_{m,s}$, η and l out of all 9 parameters have observable impact on the appearance of the scattered component. Intuitively, this is because g and α are often very close to zero in realistic fur/hair models, so they only have very subtle impact on the scattered component. The roughness parameter of unscattered lobes β is small compared with the roughness of scattered lobes, so its effect is minimal in the scattered component where each path has at least one scattered interaction. We don't use $\sigma_{m,a}$ as input because it is correlated with κ and $\sigma_{c,a}$.

On the other hand, the parameters of a dipole model include σ_a and σ_s as we introduced in Chapter 2. To enhance the flexibility of the dipole model, we add another brightness scaling parameter w, which is multiplied with the rendered dipole component at the end. Note that for the dipole model, the anisotropy parameter g and σ_s are correlated. Therefore, we only change σ_s as in the dipole model and always set g to zero. We do not have any special restriction on the value of w in order to have more flexibility in the model.

Given the high dimensionality of the parameter spaces, we use a multi-layer perceptron neural network (MLPNN) as our model for the parameter conversion problem. Figure 4.39 shows our model structure. For the input parameters, we first apply a preprocessing step to map them into an appropriate range that is easy for neural networks to train. The preprocessing function for each input is listed in the figure. Then, we feed the 5 preprocessed input parameters, $X = \{x_0, ..., x_4\}$ into a MLPNN with 2 10-node fully connected hidden layers using tanh activation function to produce 3 output parameters, $Y = \{y_0, y_1, y_2\}$. Finally, we apply a post-process to convert the output to dipole model parameters σ_a, σ_s and w as follows:

$$\sigma_a = 2^{3\tanh(y_0)+5},\ \sigma_s = 2^{3\tanh(y_1)+5},\ w = 5\tanh(y_2) + 5. \tag{4.37}$$

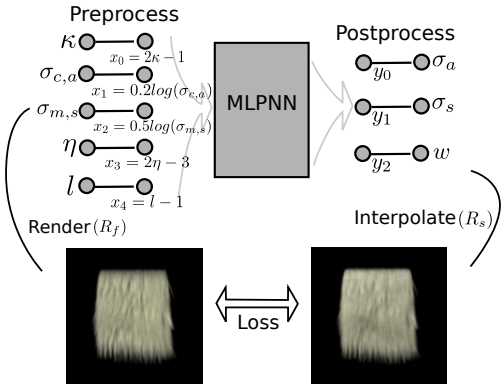


Figure 4.39: Our neural network structure for the parameter conversion problem.

Note that the mapping we choose in post-processing is to fit our training scene settings, which we will elaborate in Section 4.12.

We chose a very simple neural network structure instead of deep neural networks mainly for two reasons. First, the parameter conversion function should be very smooth given that the appearances of both models vary smoothly with changing input parameters. Therefore, using a simple structure is sufficient and avoids over fitting. Second, the neural network needs to be evaluated at each shading point during rendering to support heterogeneous fur models. Using a simple structure makes sure that the evaluation is relatively cheap compared to ray tracing. We validate in Section 4.13 that the neural network evaluation time is less than 10% of the total render time. Our neural network also has negligible memory footprint, since it only has about 200 parameters.

Training the Neural Network

To train the neural network, we need to quantitatively measure the quality of each input-output pair it generates. To do this, as illustrated in Figure 4.39, we compare the rendered image using the hair/fur model with input parameters and the rendered image using the dipole model with output parameters under the same scene settings. The resemblance between the 2 images should be a good indicator of how good the parameter conversion predicted by the neural network is.

Scene Settings. We use a single scene to train our neural network. Figure 4.39 shows an example of renderings produced by an input-output pair using the training scene. The model in our training scene is a piece of fur pelt directly facing the camera. We use the same camera and light settings and only change the material properties of the model throughout the training process. We show in Section 4.14 that although our model is only trained on one scene, it generalizes to many other scenes with different geometry, lighting, etc. This is because the parameter conversion function is not sensitive to the scene settings, and is an intrinsic property of the materials from the 2 models (hair/fur and dipole) that we try to

match. Here, we only use the training scene to find a good approximation to the parameter conversion function.

The output dipole parameters of the neural network fit the training scene well. However, if we scale the training scene by a factor, σ_a and σ_s need to be scaled accordingly to obtain correct results. In Section 4.13, we describe how to do this in scenes with different scales.

Loss Function. Suppose X is an input parameter set and $R_f(X)$ is its rendered image using the hair/fur model with only the scattered component. Recall that the unscattered component is handled separately using the traditional dual scattering in our model as introduced in Section 4.11. Therefore, we use path traced scattered components as the reference image for the dipole model to fit. $Y = f(X)$ is the output parameter set predicted by the network, where f is the parameter conversion function approximated by the neural network. $R_s(Y)$ is the rendered image using the dipole model and the output parameter set. Our loss function is defined as:

$$\text{Loss} = \gamma \text{L}_1(R_f(X), R_s(Y)) + L_s(R_f(X), R_s(Y)), \tag{4.38}$$

where L_1 is the L_1 norm between 2 images. $L_s(R_f(Y), R_s(X)) = 1 - \text{SSIM}(R_f(Y), R_s(X))$ is the structural loss term, where SSIM is the structural similarity index [136] between 2 images. γ is a parameter to weight the relative impact. In our training, we set $\gamma = 0.1$ to stress more on the structural loss term. This is because the structural loss term has much more impact on the overall visual quality than the L_1 term.

Approximate rendering using bilinear interpolation. For efficient training, R_s needs to have a fast evaluation method and be differentiable in order to do gradient back-propagation. This is extremely difficult, if not impossible, for a dipole rendering system. To solve this, we approximate R_s using a bilinear interpolation method as in Figure 4.40. Specifically, we pre-render and store images with different σ_a and σ_s on a regular 2D grid in log space, covering the potential range of the 2 parameters. Then, for a given output parameter set σ_a, σ_s and w, we first use σ_a and σ_s to perform a bilinear interpolation on the 2D grid to approximate the rendered image and then use w to scale the overall brightness of the image. This way, R_s becomes differentiable and can be evaluated efficiently through a bilinear interpolation. For the training scene, where the model is bounded in a unit radius sphere, we find that a σ_a and a σ_s both in the range $[2^2, 2^8]$ are sufficient. Therefore, we define our post-processing mapping in Equation 4.37 such that the two parameters both fall in the range. Similarly for w, a range in $[0, 10]$ is sufficient.

At this point, we have all the ingredients to train the neural network. We generated a dataset with random input parameters and corresponding rendered images. In each training iteration, we randomly select a subset of the training examples from the dataset and minimize the loss function using gradient descent. We will provide more details about our training settings in Section 4.13.

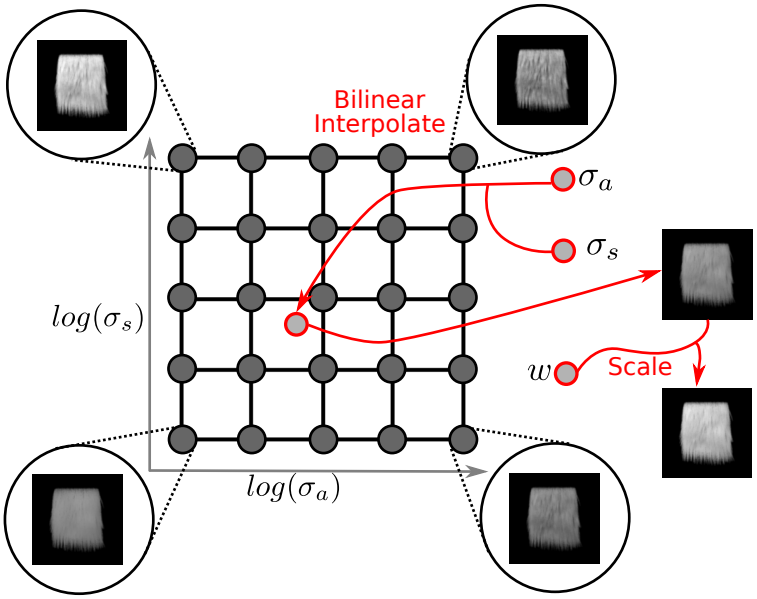


Figure 4.40: We approximate R_s using bilinear interpolation. Before training the neural network, we store a 2D grid of pre-rendered images with varying σ_a and σ_s in log *space. Given the 3 dipole parameters, σ_a, σ_s and w, we first use σ_a and σ_s to perform bilinear interpolation to obtain the appearance of the rendered image. Then, we scale the interpolated image by w to obtain the final approximated rendering.*

4.13 Implementation of Fur Global Illumination

In this section, we provide key implementation details of two relevant aspects: training and rendering.

Neural Network Settings

We generate both the training samples and the 2D grid of output rendered images in Figure 4.40 using the *Fur pelt* scene. For training samples, we rendered 1000 images in 128x128 resolution with 1024 samples per pixel using different input parameter sets randomly sampled from the 5D input space. Note that for each image, we use different $\sigma_{c,a}$ in the RGB channels to increase diversity. Figure 4.41 shows some of the examples. We list our sampling strategies for each input dimension in Table 4.9. For fixed parameters, we set β_m and β_n to 0.1 and the rest to 0. As introduced in Section 4.12, these fixed parameters do not have observable impact on scattered components. For the 2D grid of rendered images, we generate a 15x15 uniform grid in the log space of σ_a and σ_s.

We trained our neural network using Tensorflow with Adam gradient descent optimizer. Training takes about 20 minutes to converge on our data set. We obtained 0.90 structural similarity and 0.070 L1 norm on average in our training set. We also tested the neural network in a validation set, where we generated 100 images using the same sampling strategy. In the

$\kappa(0.3, 0.9)$	$\sigma_{c,a}(0.01, 4)$	$\sigma_{m,s}(0.25, 4)$	$\eta(1.2, 1.7)$	$l(0.3, 2.5)$
$0.6u + 0.3$	2^{9u-7}	2^{4u-2}	$0.5u + 1.2$	$2.2u + 0.3$

Table 4.9: Sampling strategies used for our data set generation, where u is a uniform random number in range $[0, 1]$. Please refer to Table 4.1 for the meaning of each parameter. Note that we choose these strategies to cover the range of realistic hair/fur parameters for the hair/fur model in Yan et al. [141], which we list in the parenthesis next to each parameter.

validation set, we obtained 0.89 structural similarity index and 0.072 L1 norm. Note that a structural similarity index around 0.9 suggests fairly high resemblance between two images. We show more validation of our neural network in Section 4.14.

Rendering

Parameter Scaling. The dipole model parameters we obtained in Section 4.12 is for the scale of our training scene. To generalize to scenes with different scales, we use the estimated mean free path to scale σ_a and σ_s, since these 2 parameters are inversely proportional to the mean free path. Specifically, for a given input scene, we estimate the average surface density d of the hair/fur fibers. The mean free path of the hair/fur volume is inversely proportional to $\sqrt{d}$. This is because if we scale the scene up by a factor of 2, the average distance between 2 closest hair/fur fibers also scales up by 2 and the surface density of the hair/fur fiber becomes 1/4 of the original. Suppose the training scene's estimated hair/fur fiber density is d_0, we scale both σ_a and σ_s by $\sqrt{d_0/d}$ for rendering the scene. As we show in Section 4.14, this simple scaling scheme generalizes very well with different scene scales.

Heterogeneous hair/fur parameters. To support heterogeneous hair/fur parameters, we simply perform the parameter conversion using the neural network introduced in Section 4.12 at each shading point and evaluate the dipole model using local parameters. Note that the size of our neural network structure is very small, so that the overhead of the evaluation is minimal compared to the ray tracing cost. We validate this using the hair with medulla scene shown in Figure 4.35(e). The hair scene is homogeneous, so the neural network only needs to be evaluated once at the beginning. For comparison, we render the scene with/without re-evaluating the neural network at each shading point. The render time without re-evaluation is 99s and the render time with re-evaluation is 105s. This suggests that the overhead of the neural network evaluation is less than 10% of the total render time.

Multi-bounce Illumination For multi-bounce illumination with dual scattering, we designed an importance sampling scheme for the dual scattering shading which we provide in our supplemental materials. With the importance sampling scheme, we can follow the traditional path tracing routine by recursively sampling rays and computing shading with dual scattering models. However, one issue is that strictly following the original path tracing routine would overestimate energy. This is because the dual scattering model already handles local multiple scattering in a neighborhood. This way, if the next bounce is a nearby hair

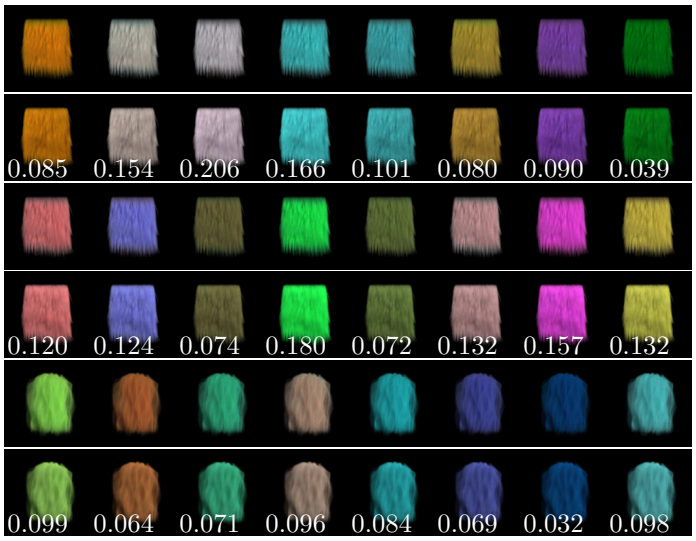


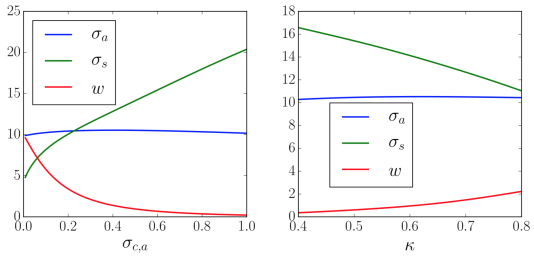
Figure 4.41: Validation of our neural network training. Odd numbered rows are rendered with path tracing of actual hair/fur fibers as reference. Even numbered rows are rendered using best fitted/predicted parameters as BSSRDF. Top two rows: selected training set and fitting. Middle two rows: selected test set with same geometry but different lighting conditions. Bottom two rows: selected test set on another scene. Our trained neural network predicts a perceptually similar match, compared to the reference. The numbers in the figures show the loss value of each fitted image. Recall that the loss function is defined in Section 4.12.

fiber, the energy would be over estimated. To solve this we set a threshold d_m for the distance between 2 consecutive bounces with dual scattering material. If the distance is smaller than d_m, we stop the recursive path tracing since the next bounce has already been considered by the dual scattering shading at the current bounce, otherwise we continue tracing the ray as usual. We apply the same strategy for the recursive irradiance sampling in the dipole model for the same reason. For all our test scenes, we set $d_m = 100r$, where r is the average hair/fur fiber radius of the hair/fur model at the current bounce point.

4.14 Results of Fur Global Illumination

In this section, we first validate our trained neural network, then show rendering results with full global illumination generated using our BSSRDF model, and compare them with previous work. We implement our model in the Mitsuba renderer [**mitsuba**], and generate all results using an Intel 6-core i7 4960X CPU, hyperthreaded to 12 threads. Upon publication, we will release the source code for both neural network training and BSSRDF model implementation, as well as our trained neural network.

Validation of the training. As a typical shape of fur growth, the *Fur pelt* scene is used to train the neural network. We first validate the trained neural network with the same



(a) Curves showing how the fitted dipole parameters vary with the absorption coefficient in the cortex.

(b) Curves showing how the fitted dipole parameters vary with the medulla index.

Figure 4.42: Parameter conversion curves learned by our MLPNN.

	Figure	#Strands	#Segs	#Samples	Time
Pelt	4.33	12.5K	4	42	7.0min
Raccoon	4.43	260K	22	64	5.4min
Wolf	4.44	1.9M	5	17	5.8min
Hamster	4.45	580K	15	31	5.0min
curly	4.46	53K	64	42	5.0min

Table 4.10: Statistics for all our scenes, rendered in 720p resolution. For each scene, there are # Strands hair/fur fibers, each with # Segs line segments. Each scene is rendered using # Samples using our 3-component global illumination model. Pre-distributing dipole samples usually takes less than 1/10 of the overall rendering time, so we do not list these timings here.

scene geometry but different lighting conditions and different fur fiber parameters. Then we validate using a new hair geometry, also with different lighting and fiber parameters, to demonstrate that it is sufficient to train on only one scene and use the result to predict other scenes. Figure 4.41 shows some of the validation results. We also include all validations in the supplemental material.

Figure 4.42 shows curves learned by the MLPNN structure. In Figure 4.42 (a), we show how the dipole parameters vary with the absorption coefficient in the cortex, while keeping other input parameters fixed. As $\sigma_{c,a}$ increases, each hair fiber absorbs more energy. This would result in a darker and sharper looking appearance of the hair model. In this case, we can see that the dipole model decreases w to match the darker appearance and increases σ_s to match the sharper look. In Figure 4.42 (b), we show how the dipole parameters vary with the medulla index, while keeping other input parameters fixed. Increasing the medulla size would produce more scattering events between hair fibers, which leads to a brighter and more smooth looking appearance. The learned dipole model responds by increasing w and decreasing σ_s in this case, which produces similar effects. Note that σ_a stays relatively flat in the learned function. This is probably because some correlations exist among the 3 output parameters. For example, increasing σ_a and decreasing w would both decrease the brightness of the rendered model. For this reason, the MLPNN probably finds it easier to fix one of

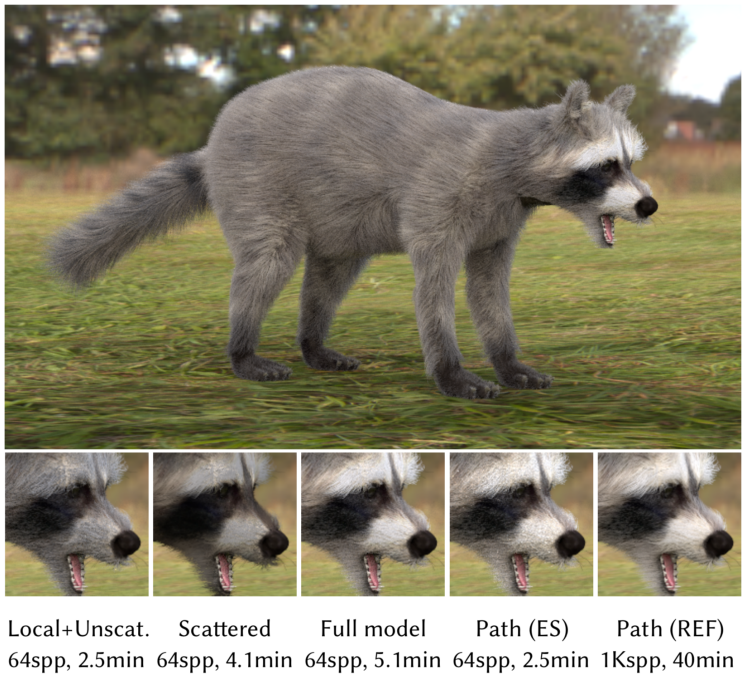


Figure 4.43: (Top row) A rendering of the Raccoon *model using our efficient 3-component global illumination model. (Bottom row) Insets showing decomposed renderings of different components, compared to path traced reference with* 64 *samples and* 1024 *samples per pixel, respectively. We find that the main source of noise is the unscattered component using dual scattering. The scattered component converges much faster. Overall, our full model produces much less noise than path tracing for equal samples.*

them in the learning process. However, we should note that the learned relatively constant values of σ_a are still crucial to the appearance of the dipole model and all 3 parameters are essential.

Scene configurations. After validation of our neural network training, we now use the trained neural network to convert parameters for actual scenes. Most of our scenes are taken from Yan et al. [141] to enable direct comparison to their method. We list all the scene configurations including geometry complexity and performance in Table 4.10. We compare the scenes with our extended dual scattering and Yan et al. [141] with path-traced global illumination as reference, and we describe them next. In all our scenes, we use 3 recursive bounces for our dipole approximation and 4 indirect samples for irradiance estimation.

Fur pelt. We render the *Fur pelt* with a blocking sphere casting a shadow onto it, and compare our rendering with dual scattering. As demonstrated in Figure 4.33, dual scattering only lights the un-blocked regions, resulting in hard shadow boundaries and no color bleeding. However, significant scattering effects are observed in the reference, and our BSSRDF model matches that much more accurately.

Raccoon. The *Raccoon* scene is rendered with a uniform sky environment map with a

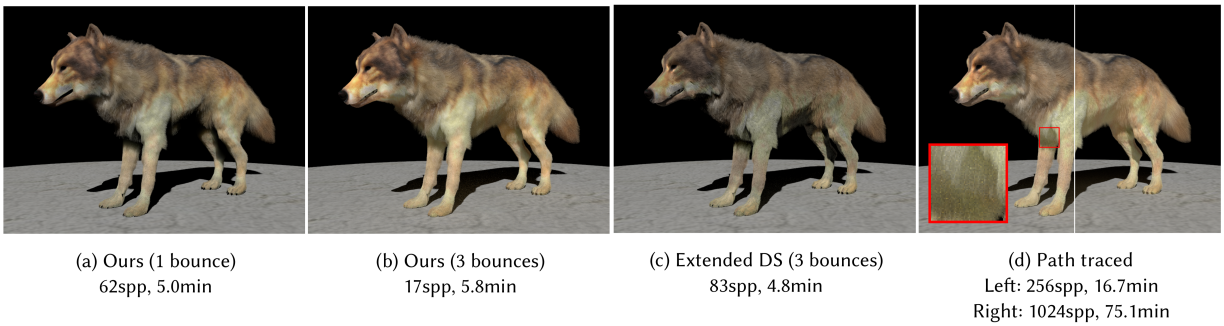


Figure 4.44: The Wolf *scene rendered using a point light. We compare our method with* 1 *bounce and* 3 *bounces with the extended dual scattering method and the path traced reference. Our method with* 1 *bounce is already better than the extended dual scattering method in terms of more accurate color and softer appearance. Moreover, our method with* 3 *bounces enables inter-reflections between fibers in the same fur volume, filling shadows on the belly and the limbs and producing a similar appearance as the reference, but still achieves a minimum of* 3× *speed up. The path tracing with* 256 *spp is still noisy, as the inset shows.*

sharp point approximating the sun. Figure 4.43 shows decomposed components and corresponding rendering time. We can see that, most energy is captured by the BSSRDF part, which soon becomes noiseless as the number of samples increases, thanks to the smoothness of the dipoles' contribution.

Wolf. The *Wolf* scene is to show our dipole model with multiple global illumination bounces. In Figure 4.44, we show side by side comparisons with and without multiple bounces using our method, rendered using a point light, and compare with the extended dual scattering, roughly for the same rendering time ($\sim$ 5 min). The extended dual scattering cannot handle multiple bounces, thus leaving hard shadows around regions near the belly and the limbs. Furthermore, it has a significant color difference with the reference, and it generates hard and solid appearance, especially on the head. Our method with 1 bounce can already capture accurate color and soft appearance. With 3 bounces, we are able to generate similar appearance with the reference. The rendering time increases roughly linearly with the number of bounces, but we are still at least 3 times faster, since the path traced result at 256 spp is still noisy when zooming in.

Hamster. The *Hamster* scene is rendered using a spot light in Figure 4.45. We also compare with the extended dual scattering method, and we can see that the dense fur fibers soon diminish global scattering from dual scattering, resulting in an overly-dark appearance. Conversely, our method with 3 bounces matches the reference much closer and is still fast. Furthermore, in the accompanying video, we rotate the camera and demonstrate that our model doesn't have to re-generate all dipole samples over frames, as long as the relative lighting condition of the model doesn't change.

Curly hair. Our trained neural network can also predict the appearance of hair. We compare renderings of *Curly hair* using our model with medullary index $\kappa = 0.36$ against

(a) Extended DS (3 bounces) 162spp, 4.9min | (b) Photon mapped Equal time, 4.8min | (c) Photon mapped Equal quality, 122.9min | (d) Ours (1 bounce) 32spp, 2.1min | (e) Ours (2 bounces) 32spp, 3.8min | (f) Ours (3 bounces) 32spp, 4.4min | (g) Path traced Top: 128spp, 4.1min Bottom: 1Kspp, 33min

Figure 4.45: The Hamster *scene rendered using a spot light. The extended dual scattering method still has the color mismatching issue. However, our method with* 3 *bounces closely resembles the path traced reference, but requires many fewer samples and is an order of magnitude faster. Significant noise can be observed for equal time path tracing when zooming in. The higher bounces result in an overhead for our method, since further bounces also require significant dipole queries. However, since the dipoles' contributions are smooth, even with the reduced sampling rate, our method still converges fast.*

(a) Extended DS 101spp, 5.0min | (b) Ours 42spp, 5.1min | (c) Path traced 4096spp, 256min

Figure 4.46: The Curly hair *scene rendered with a point light. We compare our method (b) with the extended dual scattering method (a) and Yan et al. [141] (c). Alhough the overall color/intensity is generated by the extended dual scattering, it completely fails to capture the correct color distribution. The highlight is shifted towards the bottom left. In contrast, our method is able to produce a much more accurate appearance, as compared with the ground truth, yet is still an order of magnitude faster.*

dual scattering and the path traced reference, as shown in Figure 4.46. It demonstrates that dual scattering fails to capture the actual highlight positions from scattering, and it always produces a flat shading distribution.

4.15 Summary

In this Chapter, we describe an example of innovatively revealing the world's appearance—animal fur's reflectance. We start from the observation that human hair reflectance models are not suitable for animal fur rendering, then we tackle this problem by comparing microstructures between human hair and animal fur fibers. We come up with a double cylinder model accordingly to describe the optical properties of animal fur fibers, then we improve it by deriving a simplified near field model and an efficient far field model. Finally, we solve a related global illumination problem, by approximating multiple bounces of light with subsurface scattering. We propose a novel neural network solution to connect these two different ways of rendering.

In the future, apart from modeling more types of real world appearance, we propose two high-level directions for the future of appearance modeling. First, structure determines function. It is worth exploring general mathematical methods that not only describe individual micro-structure's properties, but are also able to aggregate them to an overall appearance. Second, it is possible to combine appearance modeling with geometry modeling, so that geometries can also be range-queried, filtered and anti-aliased with level of detail. In general, we believe that appearance modeling is urgently required to achieve photorealism, and to liberate artists and graphics designers in the industry.

Chapter 5

Real-time Ray Traced Realism

5.1 Introduction

In Chapters 3 and 4, we have introduced detailed rendering and detailed appearance from microstructures. Both of these topics are intended for next-generation realism. However, even without the details, current rendering still suffers from low performance with the state of the art rendering method—Monte Carlo distribution ray tracing.

Monte Carlo ray tracing is an accurate and physically correct way to render realistic images. However, even for a simple scene with basic materials (e.g. diffuse or glass materials) and common effects (e.g. depth-of-field, soft shadows and indirect illumination), its convergence to a noise-free image is still prohibitively slow for interactive applications such as video games. With the level of details that we bring to Computer Graphics, performance is becoming even more urgent and challenging. We need real-time ray traced realism.

Fortunately, there is a considerable interest in fast adaptive sampling and filtering approaches, taking advantage of significant coherence in the intensity between pixels. In this chapter, we present our fast sheared filtering work, based on the 4D sheared filtering methods as introduced in Chapter 2. These methods perform a careful frequency analysis to determine near-optimal sampling rates for a number of different effects, such as motion blur, soft shadows and spherical harmonic/ambient occlusion [Egan et al. 2011a; Egan et al. 2011b]. While the sample count reductions are dramatic, with very few additional assumptions, these sheared filtering techniques are usually memory intensive and have high reconstruction overheads. One of the key challenges is the irregular search for samples. Even if the initial samples are stratified, they are distributed irregularly once one considers the footprint of the 4D sheared filter for each pixel. Therefore, inspite of numerous efforts to accelerate the basic sheared filtering algorithm, it remained a slow process taking several minutes per frame for reconstruction, often dwarfing the cost of even offline raytracing.

Thus, sheared reconstruction was established as a theoretically sound technique that reduced sample counts by one to two orders of magnitude. But it was not practical for fast or interactive raytracing systems, since irregular sampling and high memory usage made

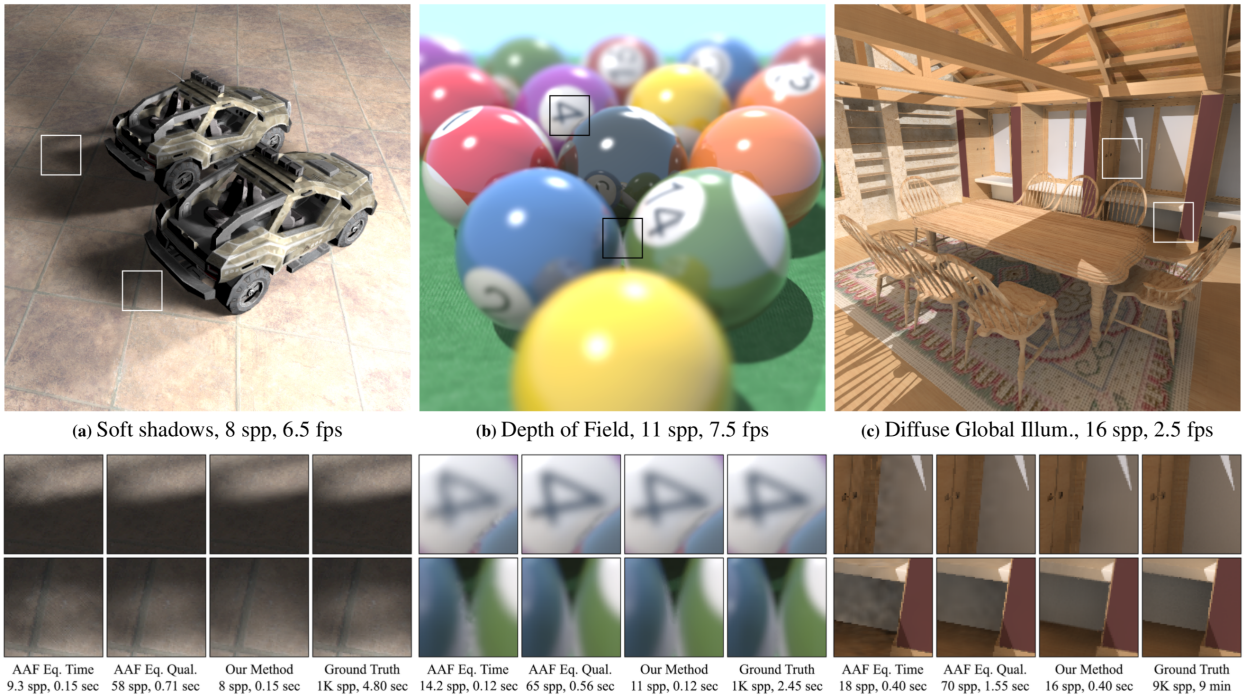


Figure 5.1: We can render soft shadows (CARS, two area lights), defocus blur (POOL, two point lights, modified from NVIDIA OptiX SDK) and diffuse global illumination (ROOM, one point light) at interactive speeds by fast 4D sheared filtering on a sparsely sampled Monte Carlo (MC) input, which is very noisy as seen in the insets. We require very low sampling rates, often under 16 samples per pixel (spp). Compared to axis-aligned filtering (AAF) with adaptive sampling [78, 79], we perform 4× faster, and reduce the sampling rate required by 5-8×.

reconstruction too expensive. Methods based on axis-aligned filtering [78, 79, 80] were developed in the past three years in response to this, to bring sampling and filtering into the real-time domain. There is a significant tradeoff in sample count, with axis-aligned filtering requiring an order of magnitude more samples than 4D sheared filtering. Nevertheless, the simplicity of the filter and its natural separability in pixel-light, pixel-time or equivalent space can be exploited to minimize filtering time, and enable inclusion in interactive raytracing systems. However, one needs many more input ray samples, since the simple filter doesn't bound the frequency spectrum tightly.

In the rest of this chapter, we describe a solution to the now long-standing problem of fast 4D sheared filtering, showing that the sample count tradeoff in axis-aligned filtering methods is no longer needed, for the common visual effects of soft shadows, depth of field, and diffuse global illumination. Indeed, we achieve the best of both worlds—the low sampling rates of sheared filtering, and reconstruction times comparable with axis-aligned filtering.

We start from the 4D pixel-light sheared filter [Egan et al. 2011b] in the primal domain for soft shadows (a similar analysis applies to depth of field and indirect illumination). Inspired by the natural separability of axis-aligned filtering, we come up with a solution

that handles the high-dimensional sheared filtering by factorizing it into lower-dimensional forms. This overcomes the problem of expensive irregular search for samples, caused by the shearing that couples pixel and light dimensions [Egan et al. 2011b]. Besides the theoretical contribution of fast high dimensional filtering, that bridges sheared and axis-aligned filtering algorithms, we dramatically reduce the practical computational cost, achieving a 4x faster implementation compared to Mehta et al. [78, 79], and orders of magnitude faster than Egan et al. [32, 31]. Specifically, we make the following contributions:

Factoring 4D sheared filter into four 1D filters: We first observe that the 4D sheared filter is a product of two 2D sheared filters along orthogonal pixel-light planes, and develop a two step factored algorithm. We then derive a further factorization into four 1D integrals, that separate the 2D sheared shape into a pre-convolution and a collection. The computational complexity[1] per pixel is reduced from $O(n^2l^2)$ to $O(nl)$, where n is the linear filter size (along one dimension) and l^2 is the number of samples per pixel (so l is the number of samples along each dimension of the lens or light). In sheared filtering, l can be very small, typically $l \leq 4$ and the samples per pixel (spp) $l^2 \leq 16$. Thus, the complexity is comparable to the $O(n)$ cost of (fully factored) axis-aligned filtering.[2]

Efficient GPU Implementation of Sheared Filter: With an efficient GPU (CUDA) implementation of the factored sheared filter, we reduce filtering time per frame to about 70 ms. In comparison, a direct implementation of the 4D sheared filter takes one to two orders of magnitude longer. This is the first *general implementation of fast 4D sheared filtering that gives interactive performance.*

Interactive Rendering of Distribution Effects: We demonstrate accurate results for soft shadows, depth of field, and diffuse global illumination with only 6-16 samples per pixel (spp), as shown in Figure 5.1. (In the main body of this chapter, we consider only single effects at a time; handling multiple effects simultaneously as in [80] is discussed briefly in the Appendix, with example images). Even though the input data is very noisy, we are able to perform high quality reconstruction. Our results match ground truth closely, which is typically obtained with 100× the number of samples per pixel (see Figs. 5.5-5.7). We implement our filtering algorithm on the GPU-based real-time Optix raytracer, and demonstrate a 4× speedup in framerate over equal quality axis-aligned filtering, while reducing sample counts by $5 - 8\times$.

5.2 Previous Work

Our work builds on a recent history of methods for adaptive image filtering to remove noise in ray traced solutions, but most of these methods were not intended for real-time use. Our

[1] The full complexity is $O(nl + l^3)$ but the $O(nl)$ term is dominant, as explained later.

[2] The fast sheared filter is still somewhat more expensive, both from the l factor, and because of the larger size n of the sheared filter. This is more than made up for, by the much smaller number of ray samples that are needed by our method, as compared to axis-aligned filtering.

approach also relates to Fourier and light field reconstruction techniques, as well as initial approaches for fast sheared filtering for depth of field and motion blur.

Image and Adaptive Filtering: Image filtering has a long history, including [105, 77]. Adaptive image sampling also has a long history, with seminal work by Mitchell [81]. Recently, Hachisuka et al. [39] presented multi-dimensional adaptive sampling and anisotropic reconstruction, that has inspired much follow-on work. Recent work also includes adaptive wavelet rendering [91], the A-Trous wavelet transform [16], cross bilateral filters [94, 92] and filtering of stochastic buffers [109]. A significant advance is random parameter filtering [108] which seeks to separate variation from random parameters and geometric signals. Other recent works are based on statistical theories like SURE [72] and non-local means filtering [103]. Recently, Kalantari and Sen [62] developed a method to locally identify noise in different parts of the image, followed by standard adaptive sampling and denoising, while Delbracio et al. [20] use ray color histograms. However, these methods do not exploit the Fourier structure of the higher-dimensional light field, and typically require high sampling rates with offline reconstruction; they are not interactive.

Real-time Distribution Effects: Real-time soft shadows are commonly produced using soft shadow maps that consider occlusion from the entire area source [37, 2]. As noted in [59], these methods make various tradeoffs of speed and accuracy. Soler and Sillion [114] provide an analytic solution, but only for geometry in parallel planes. Shadow volumes [14] can also be extended to soft shadows using geometric ideas like penumbra wedges [3] and shadow volumes [65]. Another body of work is precomputed relighting [112], but it is usually limited to static scenes lit by environment maps. Analogously, for real-time depth of field, the general approach is to rasterize layers using a pinhole camera [67, 70], and then splat and gather the samples on the image plane to approximate defocus blur for a particular focus depth. There are also simpler post-processing algorithms [96, 149] that use a single pinhole rendering and depth buffer to simulate defocus blur.

Real-time approximate global illumination techniques (a survey can be found in [102]) include voxel-based cone tracing [13] on the GPU. Point-based approaches include micro-rendering such as [101] which raytraces shading points and partitions them by k-means, and then does a final gather using GPU-based photon mapping.

Although these approaches are commonly used for their high performance, they make approximations that can produce aliasing and other artifacts. Our method is based on unbiased Monte-Carlo sampling, and can offer high-quality results with nearly the same speed.

Fourier and Light Field Analysis: Our goal is to obtain low sample counts from sheared filtering [32, 31, 30], while achieving interactive filter times comparable to axis-aligned filtering methods [78, 79]. Our method applies to any sheared filtering approach; we demonstrate soft shadows, depth of field, and diffuse global illumination, but it could be easily extended to motion blur [32].[3] We also support multiple distribution effects [80], as

[3] We do not include motion blur, since the real-time GPU Optix framework does not natively support raytracing with motion. However, it can be easily implemented within a CPU renderer such as PBRT or

we briefly discuss in the appendix.

Both sheared and axis-aligned filtering are based on a frequency analysis of the light field [9, 97, 29]. Other recent work in the area includes Fourier depth of field [115] and covariance tracing [6] that uses a covariance representation of the 5D space-angle-time light field. In terms of light field reconstruction, Lehtinen et al. [69, 68] proposed a reconstruction method for motion and defocus blur from sparse sampling of the 3D/5D (spatial position, lens and time) light field, but with a high memory and computation overhead. In general, these methods are not intended for interactive use, except for axis-aligned filtering that requires higher sample counts. In contrast, we provide accurate results with very low sample counts and interactive frame rates using fast sheared filtering.

Fast Sheared Filtering: We are inspired by Vaidyanathan et al. [126], who demonstrate a fast sheared filtering approach for defocus blur. This method was later extended by Munkberg et al. [85] to handle both defocus blur and motion blur at the same time, for which Clarberg and Munkberg [11] proposed an efficient implementation. However, these methods assume *a fixed filter for a small range of depths*, and therefore require separation of the scene into multiple layers. They use a two-step approach—first project all samples through the center of the lens to neighboring pixels, accumulate per-layer color and alpha, and then do a screen-space convolution (further separated along image axes into two passes). In contrast, we pre-convolve sampled radiance at each pixel individually along the sampling dimension, and then perform a sheared spatially-varying convolution by picking up appropriate pre-convolved samples from neighboring pixels—in a total of 4 steps. Our sheared filter implementation works for multiple distribution effects (soft shadows, defocus blur, diffuse global illumination), with no need for separating the scene into multiple depth planes. Visual comparisons are made in Figure 5.9. We also analyze the computational complexity of our method, showing how it improves on the basic sheared filtering algorithm.

5.3 Motivation

In this section, we describe the basic motivation and challenges involving factorization as a solution for fast sheared filtering.

The idea is to speed up the integrals for computing the sheared filter, similar to speed-ups obtained by factoring function transforms—for example when converting an image into basis coefficients such as spherical harmonics, fourier or wavelets. In that canonical case, an image of $N \times N$ pixels is transformed into N^2 function coefficients,

$$h(u, v) = \iint f(x, y)w(x, y; u, v)\, dx\, dy, \tag{5.1}$$

where $f(x, y)$ is the image or function on a 2D domain, $h(u, v)$ are the basis coefficients, and $w(x, y; u, v)$ are the basis functions. A direct implementation has cost $O(N^4)$. However, if

Intel's Embree.

the basis is separable along x and y as $w_x(x;u)w_y(y;v)$, we can write:

$$h(u,v) = \int \left(\int f(x,y) w_x(x;u) dx \right) w_y(y;v) dy. \tag{5.2}$$

A two-stage factored algorithm can reduce complexity:

$$\begin{aligned} g(u,y) &= \int f(x,y) w_x(x;u)\, dx \\ h(u,v) &= \int g(u,y) w_y(y;v)\, dy, \end{aligned} \tag{5.3}$$

where both steps are now $O(N^3)$.[4]

Axis-aligned filtering methods that first integrate samples and then perform image-space convolutions exploit a similar speedup. However, sheared filtering is a slow algorithm because the filter is not separable. Unlike in equation 5.2, y is not an independent variable in equation 2.25. Hence, we cannot directly separate the dimensions of the sheared filter. Also note that for different x, we have varying η_x values in equation 2.25. This prevents us from separately integrating along the y'-axis, because the filter's center y is uncertain.

5.4 Fast 4D Sheared Filtering

We now describe our fast sheared filtering algorithm in its full four-dimensional form. Our key insight is that with an appropriate factorization, the general 4D sheared filter can be made separable into a two-stage 2D integral. By further factoring these 2D integrals into 1D integrals, greater speedups are obtained. We enable interactive frame rates, with overhead not significantly different from axis-aligned filtering, but with much lower sample counts. Table 5.1 gives the computational complexity of the various steps.

4D sheared filtering

Consider the 4D form of equation 2.23, with both x and y split into two dimensions each. The sheared filtering integral becomes,

$$\begin{aligned} h(x_1,x_2) = \iiiint & f(x_1',x_2',y_1',y_2') w_x(x_1'-x_1) w_x(x_2'-x_2) \\ & w_y(y_1'-y_1(x_1,x_1')) w_y(y_2'-y_2(x_2,x_2'))\, \mathrm{d}x_1' \mathrm{d}y_1' \mathrm{d}x_2' \mathrm{d}y_2', \end{aligned} \tag{5.4}$$

where we have omitted the standard deviations for clarity.

Following the definitions in flatland, here (x_1, x_2) represent receiver space coordinates (pixel coordinates) and (y_1, y_2) represents the light space coordinate. The $w_x(x_1' - x_1)$,

[4]Further speed-ups may of course be obtained by a Fast Fourier Transform or an in-place wavelet transform, but are not immediately relevant to the sheared Gaussian filters used in this chapter.

$w_x(x'_2 - x_2)$ are (spatially-varying) convolutions for each pixel (x_1, x_2), while $w_y(y'_1 - y_1)$, $w_y(y'_2 - y_2)$ are integrals which eliminate y'_1 and y'_2.

Similar to [Egan et al. 2011b], we require that the area light is parameterized with orthogonal basis vectors, guaranteeing that (x_1, y_1) and (x_2, y_2) span orthogonal 2D subspaces of the 4D light field. For the simplicity of derivation and implementation, x_1 and y_1 are arranged as parallel, and the same for x_2 and y_2.

Solving this 4D sheared filtering integral efficiently is a long-standing problem. There are two main challenges. First, the convolution center on the y-plane (y_1, y_2) is determined by the relative deviation on the x-plane, or $(x'_1 - x_1, x'_2 - x_2)$. This indicates that y_1 and y_2 are functions of x'_1 and x'_2 respectively, making the filter non-separable between x and y. Second, the visibility function f is sampled over the entire 4D space. Unlike the 2D x-plane, which is regularly divided as a pixel grid over the output image, the 2D y-plane is continuous, over which different pixels (x_1, x_2) could (and should) sample at different locations. This means separating samples on y_1 and y_2 is difficult. Hence, neither the filters nor the samples can be easily separated.

To analyze the computational complexity, we define the image resolution in (x_1, x_2) as $N \times N = O(N^2)$, where a typical $N \sim 1000$. We define the extent of the sheared filter in $w(x - x')$, corresponding to the integrals in x'_1 and x'_2 as n, where we use $n \leq 32$. The number of light samples along y'_1 or y'_2 is small and can almost be taken as a constant, since sheared filtering works with very low sample counts. We define this as l, where typically $l \leq 4$.

As shown in Table 5.1, the input dimensionality (of f) is 4D, and the output is a 2D image. The computational complexity is $O(N^2)$ for the output, and $O(n^2 l^2)$ for the integral for each pixel. The effective complexity is thus $O(N^2 n^2 l^2)$, or $O(n^2 l^2)$ per pixel. In contrast, the spatially-varying image-space convolutions in axis-aligned filtering can be performed in $O(N^2 n)$ time or $O(n)$ per pixel, using image-space separable Gaussian filters. We will show that our final separated sheared filtering algorithm has only slightly higher $O(N^2 nl)$ complexity or $O(nl)$ per pixel instead of $O(n^2 l^2)$.

Separating into two 2D integrals

The last section showed that neither the filters nor the samples are easily separable, which would suggest that accelerating the 4D sheared filter is very difficult. Our key insight is that while separating (x_1, x_2) and (y_1, y_2) dimensions directly is not possible, we can try to separate the (x_1, y_1) and (x_2, y_2) dimensions as shown in Figure 5.2. In this section, we show how the 4D sheared filter becomes a product of two independent 2D sheared filters applied over the (x_1, y_1) and (x_2, y_2) planes respectively, and we develop a two step factored algorithm. We keep the samples and reduce their dimension at each filtering step, eliminating the dependency between y_1 and y_2 by integrating them one by one.

Step 1: We first apply one sheared filter in the $x_1 y_1$ plane, effectively evaluating the inner two integrals in Equation 5.4. Since y_1 and x'_1 are related by equation 2.25, we also

Method	Input dim.	Output dim.	Integral dim.	Complexity
4D sheared filtering	4	2	4	$\mathbf{O(n^2l^2)}$
Axis-aligned filtering	2	2	2	$\mathbf{O(n)}$
2D factoring, Step 1	4	3	2	$\mathbf{O(nl^2)}$
2D factoring, Step 2	3	2	2	$O(nl)$
Our Method, Step 1a	4	4	1	$O(l^3)$
Our Method, Step 1b	4	3	1	$\mathbf{O(nl)}$
Our Method, Step 2a	3	3	1	$O(l^2)$
Our Method, Step 2b	3	2	1	$O(l)$

Table 5.1: Computational complexity (per-pixel) and input/output dimensions of various methods. The integral dim. column is the number of dimensions we integrate over. Bold is the overall complexity (most expensive step for the factored algorithms). Here n is the linear filter size, and l^2 is the number of samples per pixel.

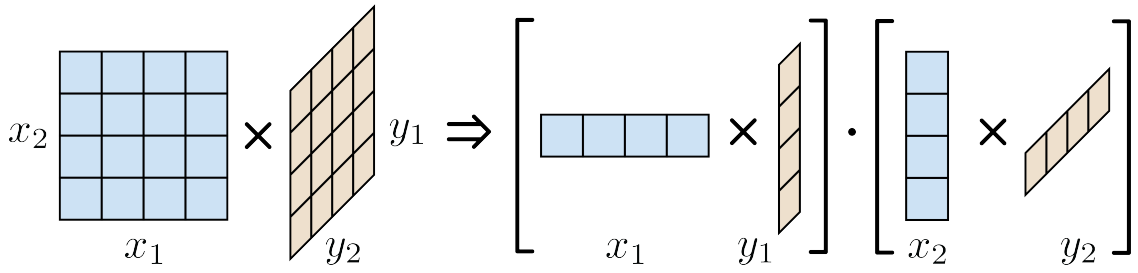


Figure 5.2: Separating a 4D sheared filter into a product of two independent 2D sheared filters in (x_1, y_1) and (x_2, y_2). Note that, as discussed in Section 5.4, separating a 4D sheared filter into 2D filters over x-plane and y-plane respectively is more intuitive, but not feasible in theory.

remove y_1. We denote the three dimensional filtered integral of f as g:

$$g(x_1, x_2', y_2') = \iint f(x_1', x_2', y_1', y_2') w_x(x_1' - x_1) w_y(y_1' - y_1(x_1, x_1'))\, \mathrm{d}x_1' \mathrm{d}y_1'. \tag{5.5}$$

Once again, we omit the standard deviations on the Gaussians for clarity. The complexity of this step is $O(N^2nl^2)$, since g needs to be evaluated at $O(N^2l)$ points, and the integral has complexity $O(nl)$. The per-pixel cost is thus reduced from $O(n^2l^2)$ to $O(nl^2)$.

Step 2: Once g is computed, we apply the second sheared filter. We integrate along x_2' and y_2' to determine the final pixel irradiance $h(x_1, x_2)$. Similarly, y_2 is eliminated since it is determined by the value of $x_2' - x_2$,:

$$h(x_1, x_2) = \iint g(x_1, x_2', y_2') w_x(x_2' - x_2) w_y(y_2' - y_2(x_2, x_2'))\, \mathrm{d}x_2' \mathrm{d}y_2' \tag{5.6}$$

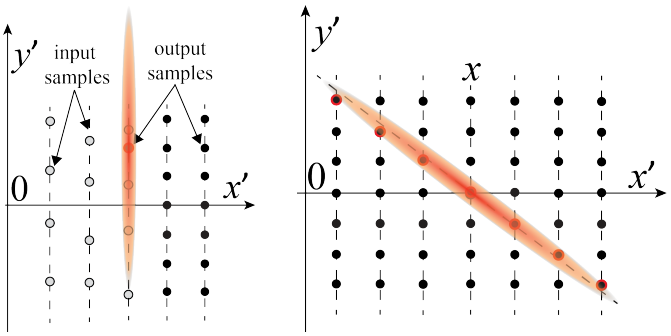
The complexity of this step is $O(N^2nl)$ or $O(nl)$ per pixel. It is less than step 1, since

we only need to produce a 2D output, and the dimensionality of g is already less than that of f.

In theory, the separation of the 4D filter into a product of two 2D filters in equations 5.5 and 5.6 is exact only when the filter kernels remain constant over the image plane. However, similar to separating a 2D box filter over an image into a two-pass orthogonal linear filter, the inaccuracy when this approximation is violated is usually negligible in practice. We will evaluate our approximation against brute force 4D sheared filtering in Section 5.6, and limitations are shown in Figure 5.10.

Separating into 1D integrals

While the separation into two 2D integrals provides savings, the first step in equation 5.5 is still expensive, with complexity $O(nl^2)$ per pixel. We derive a further factorization into 1D integrals, with a two-step computation of each 2D step. Both 2D filters have a sheared shape in the xy plane. The basic idea is to separate the sheared filter's shape into a pre-convolution and a collection as in Figure 5.3.



(a) Step a **(b)** Step b

Figure 5.3: We first separate the 4D sheared filter into two 2D sheared filters, and then evaluate each 2D filter in two 1D integration steps. As shown in (a), we first convolve along the y-axis and compute a y-dependent function (we show the visibility samples with open circles, and reconstruction locations with filled black circles, and the red circle shows an example convolution). Then in (b), we convolve along the x-axis to remove the y-dependence, effectively collecting pre-convolved samples along the shear direction, using the nearest neighbor pre-convolved values (shown in red).

Step 1a: To compute g in Step 1 efficiently, we first perform a pre-convolution for each y_1 (outer integral in Equation 5.5), to produce an intermediate result p:

$$p(x'_1, x'_2, y_1, y'_2) = \int f(x'_1, x'_2, y'_1, y'_2) w_y(y'_1 - y_1)\, \mathrm{d}y'_1. \tag{5.7}$$

There are several important points to note here. Unlike elsewhere in this section, y_1 is an independent variable, and p is precomputed (pre-convolved) for each y_1 in preparation for

step 1b, where y_1 will be expressed as usual in terms of x_1 and x'_1. p is calculated for each pixel (x'_1, x'_2) and each value of y'_2.

Since y_1 is a continuous parameter, we discretize (stratify) the range of y_1 into $O(l)$ bins.[5] In practice, accurate reconstruction requires about $4l$ bins. Since $l \leq 4$ in our case, we use 16 bins.

Note that p needs to be stored at $O(N^2l^2)$ points, and the cost of the integral is $O(l)$, so that the total complexity is $O(N^2l^3)$ or $O(l^3)$ per pixel. While this is still cubic, note that $l \ll n$ is a small constant (typically $l \leq 4$), and this cost is generally less than the $O(nl^2)$ per-pixel complexity of equation 5.5.[6] In practice, step 1a is not even the most expensive in our implementation.

Step 1b: After this pre-convolution step, we can finally compute g by applying the filter in the x-dimension (inner integral in Equation 5.5 as follows:

$$g(x_1, x'_2, y'_2) = \int p(x'_1, x'_2, y_1(x_1, x'_1), y'_2) w_x(x'_1 - x_1)\, \mathrm{d}x'_1. \tag{5.8}$$

This is a 1D integral that does a "gather" around x_1. Since p is already computed, it can be quickly queried. Note that the parameter y_1 on right, is a function of $x'_1 - x_1$ as usual, and is also integrated out in this step. Since we still need to filter and integrate g along y'_2, we also discretize y'_2 into $O(l)$ bins, similar to y_1.[7]

The complexity of this step is $O(N^2nl)$ since we need to store g at $O(N^2l)$ values, and the integral has complexity $O(n)$. The per-pixel cost is thus $O(l^3 + nl)$ from combining steps 1a and 1b. In practice, the sheared filter size n is about 32 pixels, while the number of samples on the light l^2 is usually about 16. Hence we have $n > l^2$ and step 1b dominates, with the net per-pixel complexity being $O(nl)$. *Note that this is the square root of the original $O(n^2l^2)$ complexity in equation 5.4.*

Step 2a: As for step 1, we separate the sheared filter in step 2 into two 1D filters. Similar to step 1a, we first pre-filter the result of step 1b, and determine the result q for every possible y_2:

$$q(x_1, x'_2, y_2) = \int g(x_1, x'_2, y'_2) w_y(y'_2 - y_2)\, \mathrm{d}y'_2. \tag{5.9}$$

Step 2b: Finally, we integrate on x'_2 while quickly querying q,

$$h(x_1, x_2) = \int q(x_1, x'_2, y_2(x_2, x'_2)) w_x(x'_2 - x_2)\, \mathrm{d}x'_2. \tag{5.10}$$

This last step also integrates out the y_2 dependence, because y_2 depends in the usual way on $x'_2 - x_2$. Finally, we have evaluated the 4D integral of h in equation 5.4 using four 1D filters.

[5] In practice, we use uniform jittered sampling so the samples for a given pixel are offset the same way in each stratum, but this is not critical for our method as long as stratified sampling is used.

[6] Just as in practice we must use $\approx 4l$ bins, we need a similar number of bins for storing y'_2 in equation 5.5, since the jitter offsets for different pixels are different for the next step.

[7] Instead of enumerating each possible y'_2 value and searching for feasible p samples, we instead use an inverse method by projecting different y'_2 values of the p samples onto the discretized y'_2 space.

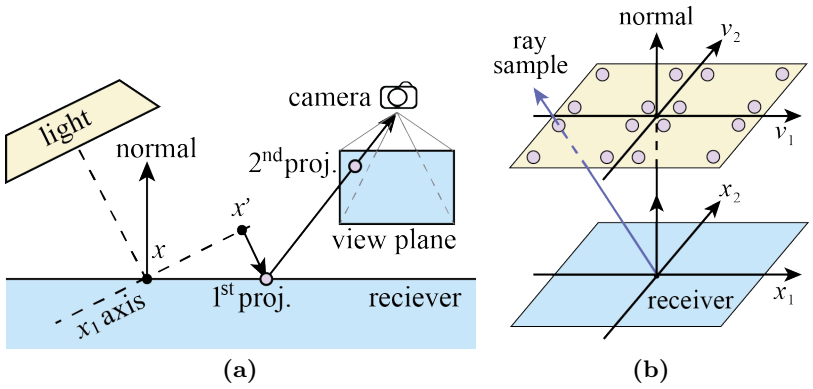


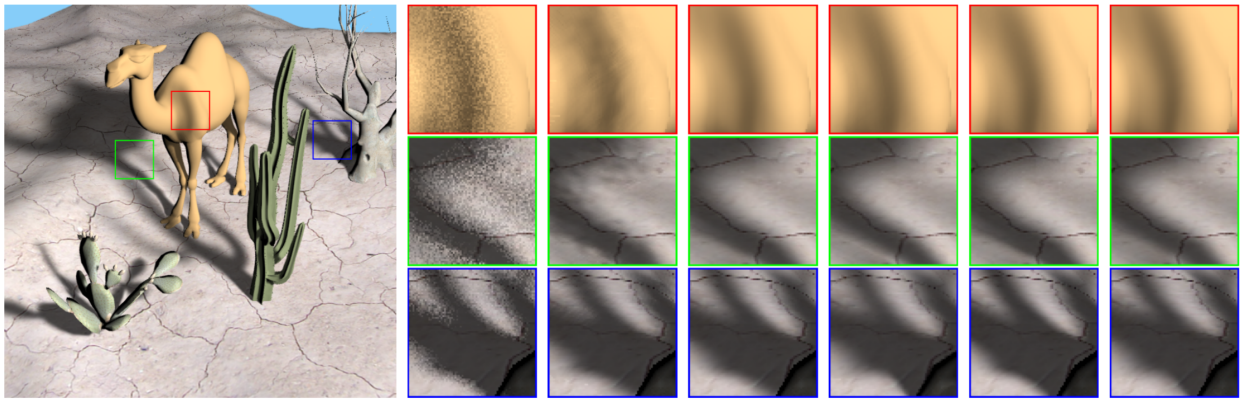
Figure 5.4: (a) Since soft shadows must be filtered along x_1, x_2 axes as defined by the light source, we first determine these axes in screen space, by projection from the light to the receiver and then to image space. (b) We sample indirect illumination along the v-plane instead of the usual cosine-hemisphere sampling.

The complexity of step 2a is $O(N^2l^2)$ and of step 2b is $O(N^2n)$. Since $n > l^2$, the last step dominates and the net per-pixel cost is $O(n)$ per pixel, as compared to $O(nl)$ for equation 5.6. Step 1b is the overall dominant cost, which is $O(nl)$ per pixel. Since l is a small constant, this is effectively $O(n)$ per pixel, *and the computational complexity is comparable to (but with higher constants than) axis-aligned filtering.*

5.5 Implementation

Our implementation involves two basic components, as in most previous work: Sampling by ray or path tracing to obtain the original noisy samples of $f(\cdot)$, and filtering or reconstruction by fast sheared filtering. We implement our algorithm using OptiX 3.0 and CUDA 5.0 interoperation. We use OptiX to do the sampling step, and we store the result in OptiX buffers. Then we use four sequential CUDA pixel shader passes to perform our four-pass filtering. We will release the source code online upon publication. Note that while we have described filtering using integrals above, these map almost directly to discrete summations over a grid of points. We now discuss a few important details of the implementation.

Ray Tracing and Sampling: To reduce memory footprint while avoiding banding artifacts, we use uniform jittered sampling [98], so that a given pixel's samples have the same random offset on a regular stratifed sampling grid. This makes it easier for the filtering steps, that can now operate on a regular grid, as well as in reducing memory in storing samples for intermediate stages. For soft shadows and depth of field, we sample uniformly on the light and lens respectively (and use Gaussian filter weights which account for the Gaussian intensity and aperture respectively). For a given pixel, we store the jitter value ($\in [0,1]^2$), and each sample's visibility or radiance, i.e., discrete samples of the function $f(\cdot)$ for filtering.



(a) Soft Shadows (Ours), 9 spp, 12.2 fps
(b) MC Input 9 spp, 0.05s
(c) AAF (ET) 13 spp, 0.08s
(d) AAF (EQ) 46 spp, 0.30s
(e) Ours 9 spp, 0.08s
(f) Gr. truth 1K spp, 3.94s
(g) 4D SHF 9 spp, 2.92s

Figure 5.5: The CAMEL scene with soft shadows, rendered at 12.2 fps with 9 samples per pixel (spp), demonstrates our ability to accurately reconstruct overlapping and thin-occluder shadows. Comparisons show (b) noisy unfiltered MC input to our method (note that we store individual samples for sheared filtering), (c) Equal time (ET) Axis-aligned filtering (AAF) retains some low-frequency noise, and overblurs sharp shadow edges (since we use $\mu < 1$) (d) Equal time (EQ) AAF is 4× slower, while (g) simple 4D sheared filtering is 50× slower.

			Sampling	Our fast GPU sheared filtering algorithm						AAF	Total	
Scene	Tris	spp	Optix (ms)	Pre/Post filt.(ms)	Step 1a (ms)	Step 1b (ms)	Step 2a (ms)	Step 2b (ms)	overhead (ms)	AAF (ms)	time (ms)	fps
CARS	4 K	8	85.2	8.2	4.0	28.6	11.4	16.2	**68.4**	32.0	153.6	**6.5**
CAMEL	43 K	9	48.0	4.3	2.0	14.4	5.7	7.6	**34.0**	16.0	82.0	**12.2**
POOL	-	11.0	69.5	1.2	4.8	25.8	11.3	20.6	**63.7**	13.0	133.2	**7.5**
STILL LIFE	233 K	11.2	146.0	2.0	6.4	44.0	11.1	22.4	**85.9**	18.0	231.9	**4.3**
ROOM	163 K	16	310.0	-	7.5	47.9	11.5	23.8	**90.7**	75.0	400.7	**2.5**
SIBENIK	75 K	16	225.2	-	7.5	40.8	11.5	23.0	**82.8**	72.5	310.0	**3.2**

Table 5.2: Detailed timings of our scenes (in milliseconds) rendered at 720×720. Cars and Camel show soft shadows, Pool and Still Life are depth of field, Room and Sibenik are diffuse global illumination. We list triangles and samples per pixel for all six scenes (Pool uses spheres rather than triangles). We also list the per-frame sampling time for raytracing in Optix, followed by timings for various stages or our algorithm, and the total overhead for fast sheared filtering. For comparison, we also list the total overhead for axis-aligned filtering. Finally, we list the total time and frame rates. We achieve interactive frame rates of 3-12 fps on a variety of complex scenes.

These samples also determine the pixel's frequency information, and ultimately the slope bounds for the sheared filter, $s_{\min}$ and $s_{\max}$. In contrast to previous work on axis-aligned filtering, we use only a single (non-adaptive) sampling pass, since our sampling rates (4 to 16 samples per pixel) are so low; these samples determine both $f(\cdot)$ and the sheared filter.

Filtering: Our filtering algorithm was described in Section 5.4. We clamp the maximum filtering range to a diameter of 32 pixels; this prevents rapid changes in rendering speed when the view is changed. The number of discrete bins used in filtering steps 1a and 1b for storing

y values is $16 \approx 4l$. In steps 1b and 2b, it is required that we filter along x_1, x_2 directions exactly—these are given by the projection of the light's axes y_1, y_2 on the receiver for soft shadows. So, we first compute each pixel's x_1 and x_2 in the world coordinate frame, sample along each direction, then project the sampled point back to the screen as shown in Figure 5.4(a). For indirect illumination, the filtering directions x_1, x_2 are orthogonal in world space to the receiver normal, and locally aligned. For depth of field x_1, x_2 are exactly along the screen's row and column axes, so the filtering algorithm can be applied directly.

Slope Smoothing for Soft Shadows: After the initial sampling, many pixels on the edges of shadows do not have valid $s_{\min}, s_{\max}$ values (if none of the samples hit occluders), which causes edge artifacts. Hence, similar to [78], we obtain the slope range for unoccluded pixels by smoothing over a 5×5 window. The (small) time for this operation is shown as pre-filtering in Table 5.2, and included in the total overhead of our algorithm.

Adaptive Sampling for Depth of Field: For depth of field, for some pixels, the slope bounds $s_{\max}$ and $s_{\min}$ could be of opposite signs. In such cases, the shear value can be close to zero, and the filtering is inaccurate. As in [126], we find that using axis-aligned filtering gives better results for pixels with $s_{\max} \cdot s_{\min} < 0$. These pixels usually need more samples even with filtering, so we trace a fixed 36 more samples for such pixels in a second sampling pass (and then do standard axis-aligned filtering—this small post-filtering time is reported as part of the overhead in Table 5.2). In most cases, the fraction of the image that requires further sampling is very small. For the scene of Figure 5.6, the first pass requires 9 spp and the average from the second pass is 2.2 spp.

Sampling/Filtering for Indirect Illumination: As described in Chapter 2, indirect illumination must be filtered in v-space, and hence we also sample on the v plane. For a single pixel, we map a uniform jittered sample with a cubic function and a scaling, to approximate the non-uniform PDF of the diffuse transfer function $\gamma(v_1, v_2)$ on $(v_1, v_2) \in [-5, 5]^2$, and then compute the ray direction, as shown in Figure 5.4(b). Theoretically, the range of (v_1, v_2) is the infinite plane, but our truncation contains over 99% of the total energy of $\gamma(\cdot)$ and only introduces a very small bias. Note that we apply the BRDF weight to each sample (v_1, v_2) and also account for the sampling PDF. Finally while filtering in steps 1a and 2a, we use a box function instead of a Gaussian as for soft shadows and depth of field, since the BRDF transfer function is no longer a Gaussian. Steps 1b and 2b (spatial filtering) can still use Gaussian weights.

5.6 Results

Our results are produced on an Intel 6-core 3.6GHz i7-4960X CPU, with a NVIDIA GTX Titan video card. We show results for interactive soft shadows in Figs. 5.1(a) and 5.5; depth of field in Figs. 5.1(b) and 5.6; and diffuse indirect illumination in Figs. 5.1(c) and 5.7. We compare to stratified Monte Carlo sampling without filtering, unaccelerated 4D sheared filtering [Egan et al. 2011b], and to axis-aligned filtering [78, 79]. The accompanying video shows animations and screen captures with moving light source, viewpoint and some

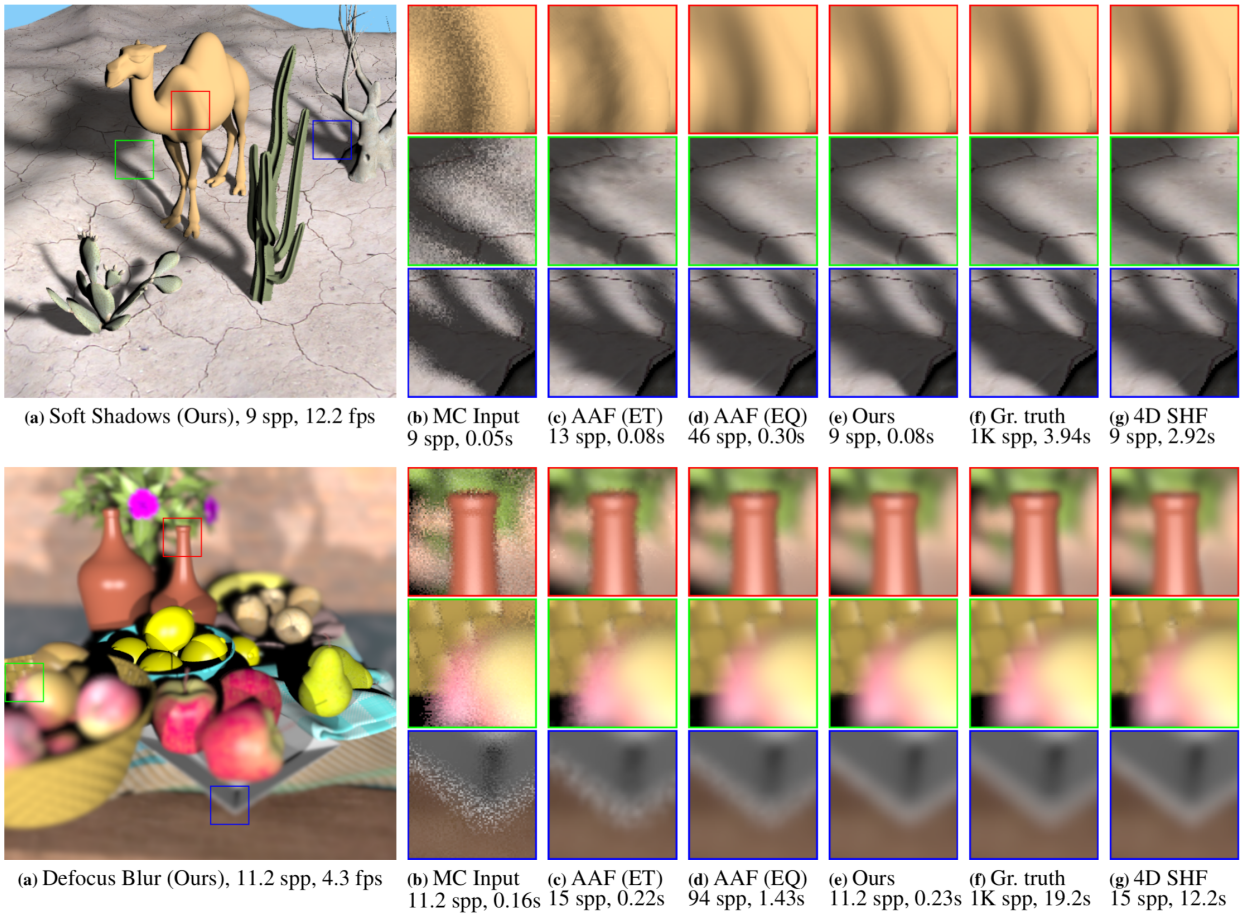
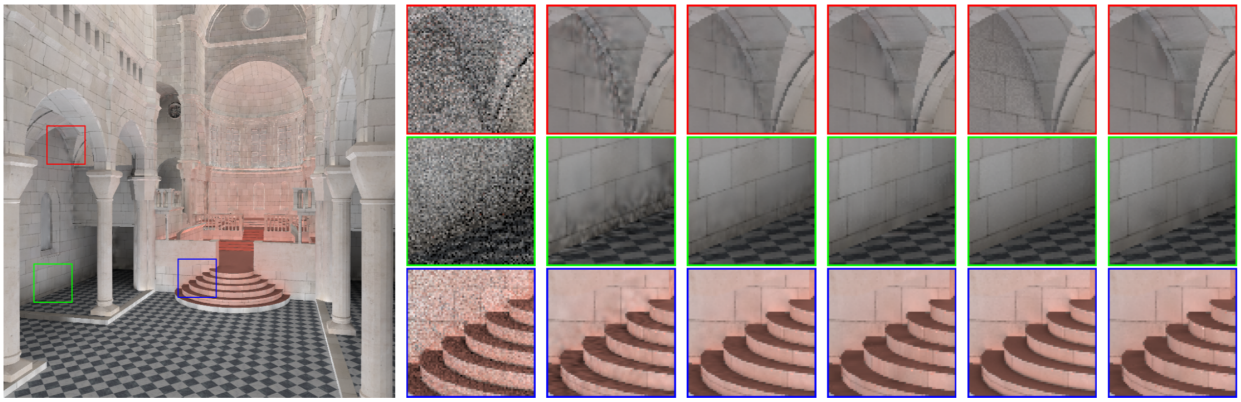


Figure 5.6: The STILL LIFE with depth of field is illuminated by two point lights and rendered at 4.3 fps with only 11.2 average samples per pixel (spp). Comparisons show (b) noisy unfiltered MC input to our method, (c) Equal time (ET) Axis-aligned filtering (AAF) retains some low-frequency noise, (d) Equal quality (EQ) AAF is 6× slower, while (g) simple 4D sheared filtering is 45× slower. Our method (and 4D SHF) produces slight overblur for background regions and underblur for foreground ones, and transition artifacts near the focal plane where AAF and our method switch. However, our method reduces most noise and requires the least number of samples.

examples of dynamic geometry. We require no precomputation except the ray-tracer BVH, and each frame is rendered independently.

Accuracy and Speedup over Monte Carlo

The accuracy of our method, and the benefit of filtering over stratified Monte Carlo is evident from the figures, for all three visual effects (soft shadows, defocus, diffuse global illumination) in a number of different situations. We take as input a Monte Carlo result with 4-9 average samples per pixel for depth of field and soft shadows, and 16 samples per pixel for indirect



(a) Diffuse Global Illum. (Ours), 16 spp, 3.2 fps | (b) MC Input 16 spp, 0.23s | (c) AAF (ET) 19 spp, 0.32s | (d) AAF (EQ) 94 spp, 1.39s | (e) Ours 16 spp, 0.31s | (f) Gr. truth 4K spp, 128.2s | (g) 4D SHF 16 spp, 7.1s

Figure 5.7: The SIBENIK scene showing only 1-bounce diffuse indirect lighting with one point light, rendered with only 16 samples per pixel at 3.2 fps. Monte Carlo input in (b) is noisy, while equal time axis-aligned filtering in (c) has artifacts at this low sample count. Equal quality AAF in (d) requires 6× as many samples as our method in (e), and is 4× slower. While there are a few artifacts remaining, our method significantly reduces noise to the level of ground truth in (f) and simple 4D sheared filtering in (g), but is more than an order of magnitude faster.

illumination. As shown in the insets of Figure 5.1, and Figs. 5.5(b)-5.7(b), this input is very noisy, but our fast sheared filtering technique produces visually accurate results compared to ground truth Monte Carlo with 1024-4096 samples, which is 100-200× slower. A quantitative comparison is in the graph of Figure 5.8. While the quantitative errors are somewhat higher than from a visual comparison, our method converges with more samples, and for equal RMS error, reduces the number of samples needed by over an order of magnitude compared to Monte Carlo, and about 6× relative to axis-aligned filtering.

Timings

In Table 5.2, we show timings for steps of our algorithm on different scenes, rendered at a resolution of 720×720. The CARS scene in Figure 5.1 has 4K triangles with soft shadows from two area lights, each sampled with 4 samples per pixel (total 8 spp). The CAMEL is a more complex example of soft shadows, rendered with 9 spp for the light. The POOL (which uses spheres, rather than triangles as primitives) and STILL LIFE show depth of field effects, while ROOM and SIBENIK are complex scenes with over 10^5 triangles, that demonstrate diffuse indirect illumination with 16 spp. The raytracing time using OptiX varies with scene complexity, from 39ms for POOL to 310ms for ROOM. The total overhead of our method is about 40-60ms for soft shadows and depth of field, and about 80ms for indirect illumination. This is less than the cost of raytracing in most cases, and about 25% of the total cost for the more complex ROOM and SIBENIK scenes, resulting in only a modest decrease in the overall performance of the real-time raytracer. Step 1b, involving the gather operation is the

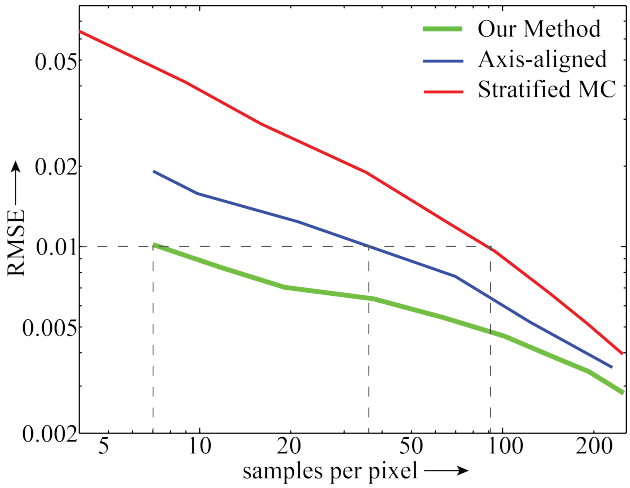


Figure 5.8: RMS error of the CAMEL scene as a function of sampling rate for our method, unfiltered stratified Monte Carlo and axis-aligned filtering with adaptive sampling. At very low sample counts, we obtain an overall benefit of more than an order of magnitude over Monte Carlo (even better visually), and around 4× over axis-aligned filtering. Moreover, our method converges to ground truth, and is always more accurate than axis-aligned filtering.

most expensive step, as discussed in the text, accounting for about half the total overhead.

Compared to axis-aligned filtering (AAF), our overhead is about 2× as much for soft shadows, 4× for depth of field, and only 20% more for diffuse indirect illumination. While equal time AAF can use slightly more samples (and adaptively sample), the many fewer samples needed by our method provides a net win of 4× in wall clock time and about $5-6\times$ sample count reduction for equal quality, as shown in the figures. Our algorithm does require more memory compared to the axis-aligned approach, since we need to store intermediate results. However, since we usually need low sampling rates, GPU memory is usually adequate. Even for visually indistinguishable convergence, we usually need no more than 25 samples per pixel for soft shadows and depth of field effects, and 49 samples per pixel for global illumination. We achieve interactive frame rates of 3-12fps for a wide range of scenes.

Comparisons

Axis-aligned filtering: We use the authors' code for soft shadows and indirect illumination, and implement depth of field analogously. Their user-specified parameter μ is used to control filter size and adaptive sampling rate. We use $\mu = 1$ for equal quality comparisons in all six of our scene figures, and this requires about $5-6\times$ the sampling rate of our method. For equal-time comparisons in Figs. 5.5(c), 5.6(c), 5.7(c), we need to use $\mu < 1$ to reduce the sampling rate. This increases the filter sizes, and slightly overblurs the image. Low-frequency noise is also retained in high-variance regions. For diffuse global illumination, much higher

sample counts are needed, and the equal time comparison in Figure 5.7(c) shows artifacts. Our method performs better; While inaccuracies exist in out-of-focus regions and artifacts can be seen around discontinuous geometry in soft shadows and diffuse global illumination situations, our method filters out most visible noise with significantly fewer samples.

Layered light field reconstruction: We use the source code of LLFR [126] to make comparisons with our method both in terms of quality and speed for filtering depth-of-field images. We use the GPU implementation of LLFR, with their default filter width of $n = 16$ pixels, rather than $n = 32$ as used in our results. Since the LLFR code takes as input only uniformly sampled light fields, we disable adaptive sampling and use a constant 9 spp for both depth-of-field scenes compared. LLFR requires segmentation of the scene into depth layers, where the same filter can be applied within a layer. Our method makes no such assumptions. As shown in Figure 5.9, LLFR produces more noise in some regions, because the sheared spectrum it is using is not as compact as ours due to layering. However, LLFR has a slightly lower reconstruction time than our method, which is partly due to their use of half-precision floating point numbers for all stored data.

4D sheared filtering: Our method is equally accurate as the full 4D brute-force sheared filter of [Egan et al. 2011b] (compare Figs. 5.5, 5.6, 5.7 (c) and (g)). We implemented the simple 4D sheared filter without any of our factorizations in CUDA, so we could compare using the same framework as our method. We used a single filtering pass for the 4D sheared filter, which accumulates radiance from all samples in the neighborhood of a pixel. As seen in the figures, this is about $40 - 50\times$ slower than our approach. For all the depth of field results using 4D sheared filtering, adaptive sampling is used according to the general sampling rate formula for sheared filters derived in [32]. At in-focus regions, since the shape of the spectrum is no longer a double wedge, 4D sheared filtering falls back to brute force Monte Carlo as described in [32].

5.7 Discussions and Limitations

Complexity: The actual complexity of our four-step filtering should be $O(l^3)+O(nl)+O(l^2)+O(n)$. Clearly, steps 1a and 1b have cost $O(l^3+nl)$ which is l more than steps 2a and 2b. In practice, $l^2 < n$, since l is typically 4 or less and n is 32. Therefore, step 1b dominates. Furthermore, since the $O(l^3)$ step 1a happens within each pixel, an efficient pixel-level parallelized implemention is used to avoid much of the overhead caused by conflicts in pixel-access. Hence, the $O(l^3)$ term can be absorbed into the numerical constant, and the total complexity would be $O(nl)$ per pixel.

Multiple Effects: Our method currently focuses on seperated single effects. Simultaneously handling multiple effects [80, 85], including soft shadows, depth of field and motion blur, requires filtering higher dimensional (often 5D or 6D) data, while the spectrum of each slice of the data for a single effect still remains a sheared shape. This can be difficult, since it further increases dimensionality and non-separability of samples for different effects. How-

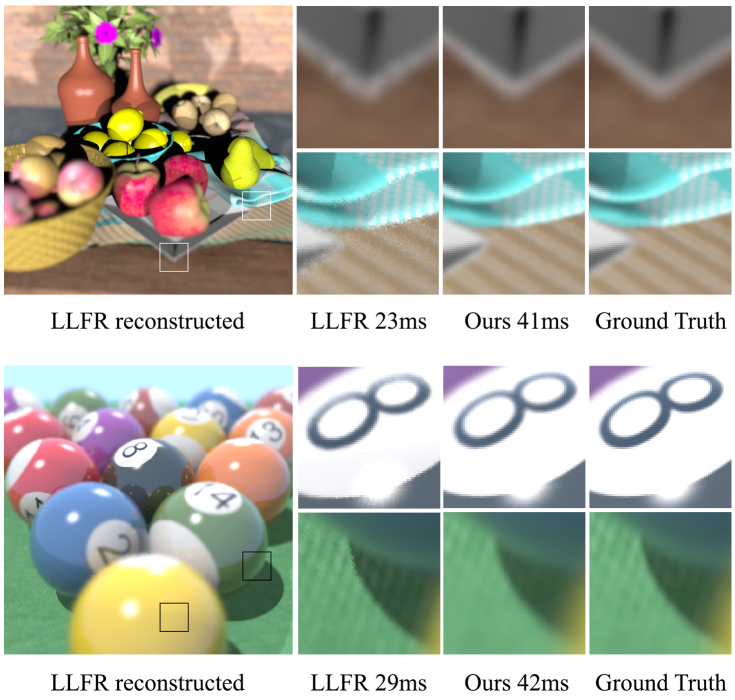


Figure 5.9: We compare insets of the STILL LIFE scene (top row) and the POOL scene (bottom row) to LLFR [126], with a uniform 9 spp for both scenes and 12 layers for LLFR (the default). Overall, LLFR has good quality and takes less reconstruction time than our method. However, it usually produces more visible noise, as mentioned in Section 5.6, and is limited to depth of field only. Our method makes no assumptions about depth layers and is general enough for soft shadows and indirect illumination as well.

ever, by introducing a reasonable approximation, we demonstrate that multiple effects could be separated into a combination of several individual effects. Please refer to the Appendix for detailed derivations and results.

Limitations: For soft shadows, minor artifacts could emerge due to projections of world space receivers to screen space, when the receiver becomes almost perpendicular to the viewing direction (projection from receiver to screen space collapses to a single point), or when the light becomes normal to the receiver (the x_1, x_2 axes become parallel). For depth of field, inaccuracies may result when the slope range at a pixel is large. In diffuse indirect lighting, a small bias is introduced in sampling due to using truncated v-plane sampling, but the bias is usually not perceivable.

When a large filter is applied to locations where the filter sizes vary rapidly, inaccuracies could occur since our method uses separable passes. This is often encountered when filtering near in-focus regions for depth of field effects, resulting in slight overblur around these regions. We do not filter between neighboring pixels if they are distant in world space, or if they have very different normals (angle threshold 20°). Pixels at which many such neighboring pixels are rejected may retain noise or artifacts. Figure 5.10 shows some of these difficult regions

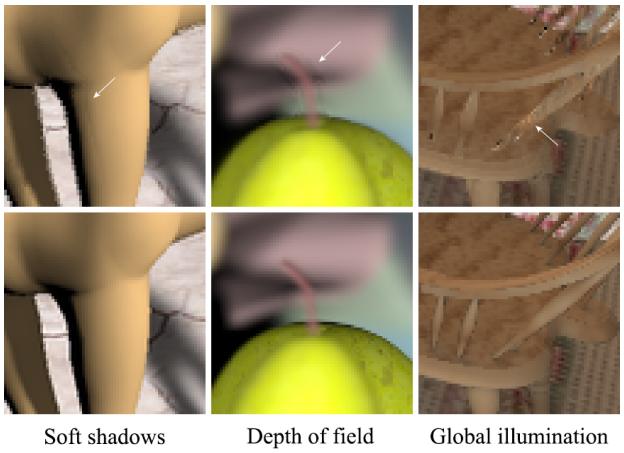


Figure 5.10: Comparisons of our method (top row) and the ground truth (bottom row) in difficult regions. Our method produces minor artifacts in certain regions as pointed out and discussed in Section 5.7.

and points out the artifacts.

In depth of field rendering, our method falls back to axis-aligned filtering in regions where $s_{\max} \cdot s_{\min} < 0$. Therefore, ghosting artifacts may appear on the boundary where this switch occurs, e.g., around the stem of the pear in the Still Life Scene (Figure 11 middle) and the topmost apple. These transitions and ghosting artifacts also occur in the original 4D sheared filtering, since the problem is not related to separability.

Our method also suffers from the general problem of sheared filtering — noisy occluding geometry. Since we are using a fairly low number of sample rays, the occluding geometry may not be accurately captured — thus the shape of the sheared filter itself could be inaccurate and noisy. This will consequently lead to flickering between frames. However, video comparisons show that our method still performs better than previous methods, even when the previous approaches use many more samples and more time.

Similar to [32], our method needs to store and filter the entire 4D light field $f(x_1, x_2, y_1, y_2)$, which is of complexity $O(N^2 l^2)$. This introduces significant storage overhead, practically limiting our method to about 25 spp for a resolution of $2K \times 2K$ (for our specific hardware configuration). For HD applications, a block-wise sampling and filtering configuration could be derived from our work with an expected but small additional performance cost for inter-block operations. We leave this for future work.

5.8 Summary

We demonstrate an interactive GPU-based method of sheared filtering for Monte-Carlo rendering of distribution effects. We propose a novel factorization of the 4D sheared filter into two 2D filters, and we further split each 2D filter into two 1D filters. We show that our method is $4\times$ faster than axis-aligned filtering for the same quality, with a $5-6\times$ reduc-

tion in sample count. We believe that we have taken an important step towards real-time physically accurate rendering, and expect many future developments that enable both low sampling rates and high performance.

Chapter 6

Conclusion and Future Work

In this dissertation, we aim at providing mathematically and physically correct solutions to the two ends of the rendering spectrum—realism and speed. We model and render visual appearance at real world complexity, while exploring theory and practical algorithms to make it real-time.

In Chapter 3, we reveal the fundamental relationships between high-resolution specular surfaces, small light sources, complex normal distributions and glints. These factors lead to our detailed rendering together, producing important material appearance phenomena that received minimal attention in previous research. We explained the failure of traditional Monte Carlo approaches at reproducing this effect, and introduced a new deterministic approach for computing the underlying integrals. Our key idea is to shade a surface patch seen through a pixel by evaluating the true normal distribution function of the patch for a single normal, which can be done under Gaussian kernel assumptions. The problem leads to integrals of bivariate Gaussians over triangles, which can be efficiently approximated. We showed complex, temporally varying specular reflections from materials such as bumpy plastics, brushed and scratched metals, metallic paint and ocean waves.

We then extend our algorithm for efficient rendering. We represent a surface as a *position-normal distribution*, then approximate this 4D distribution as a mixture of Gaussian *elements*. Combined with a Gaussian query for a given surface footprint and half-vector, this formulation admits an efficient closed-form solution, which can be accelerated using a 4D bounding box hierarchy. Our improved method is fast enough to be treated as a standard BRDF in a typical Monte Carlo path-tracer, along with the benefits of supporting multiple importance sampling, environment maps, and area lights (including ones with HDR emission textures). And our improved method will enable high-quality specular microstructure rendering to be integrated in practical systems.

Finally, we take this process to a new level, introducing the first practical technique for simulating full diffraction effects under wave optics in completely arbitrary micron-scale height-field geometry. The result is a dramatic change in the predicted BRDF due to reflection from small areas of surface: the unrealistically sharply defined structures of geometric optics give way to softer results that depend on wavelength, introducing color into the

BRDF. In practice, the new model produces softer, more natural looking reflections from microgeometry, with subtle color effects visible under sharp lighting.

An exciting extension of our work is that it provides the ability to represent surface features at all scales with the appropriate type of model: large features can be handled with geometry; smaller ones down to a fraction of a millimeter can be represented using geometric normal maps; and features down to wavelength scale are represented as diffracting height fields. Smaller features than that are not optically relevant and are not needed in any visual simulation. It would also be interesting to bring our approach closer to interactivity with further approximations. An extension to displacement maps would be possible as well. We could also explore related glinty phenomena caused by refraction, seen e.g. in snow, hair, waterfalls, fabrics or plant cellular structures.

In Chapter 4, we first present a physically-accurate fur reflectance model to accurately capture the appearance of animal fur fibers. Our model treats fur fibers as double cylinders, taking into account the existence of scattering medullas inside. We then derive a fast evaluation algorithm for rendering, built upon precomputed empirical medulla scattering profiles. We demonstrate that our rendering model fits the measured data well, with errors comparable to a full volumetric simulation. We also introduce the first database in computer graphics of reflectance measurements on a number of animal fur fibers, including both raw 2D scattering profiles and fit parameters. We show that our model is capable of generating a variety of realistic animal fur appearances, with significantly more realistic results than previous methods.

We then extend our first fur reflectance model for efficient fur rendering. By unifying the IORs of cortex and medulla, we simplify our model so that it is capable of representing complex scattering within hair and fur fibers with only 5 lobes. By introducing medulla's absorption and different longitudinal and azimuthal roughness, and using tensor approximation to minimize the storage overhead, our model further achieves both accuracy and practicality. Along with our improved model, we propose an analytic integration scheme for efficient far field appoximation, and extend it to handle multi-scale rendering for the first time.

In addition to our reflectance model for individual fur fibers, we present the first scattering model to efficiently approximate global illumination within hair and fur volumes. We analyze and point out failure cases of classic dual scattering, and organize our model as three components: direct illumination, dual scattering for unscattered lobes, and BSSRDF for all other scattering events. We convert properties from hair and fur fibers to BSSRDF parameters by training a neural network on only one scene. Our model supports various lighting conditions including environment maps, and enables hair to hair inter-reflections. We show close matches of our predicted renderings using the dipole method, compared with the path traced reference from Sections 4.8 and 4.9.

In summary, with our reflectance model and scattering model, we have presented a complete state of the art solution to hair and fur rendering. In the future, an extension to handle more features such as irregular fiber sections, complicated cuticle scale arrangements and discontinuous medullas would be interesting. Introducing explicit eccentricity or irreg-

ular shaped azimuthal sections would also help. And an artist-friendly perspective for our models can benefit the industry. Another practical direction is to support SRBF lights [100, 140] to enable real-time hair and fur rendering under environment illumination.

In Chapter 5, we demonstrate an interactive GPU-based method of sheared filtering for Monte-Carlo rendering of distribution effects. We propose a novel factorization of the 4D sheared filter into two 2D filters, and we further split each 2D filter into two 1D filters. We also derive a complexity analysis for our method, and compare it to axis-aligned filtering. Our results show soft shadows, depth of field and diffuse global illumination at interactive speeds for complex scenes, and we are $4\times$ faster than axis-aligned filtering for the same quality, with a $5-6\times$ reduction in sample count.

We believe that sparse sampling, followed by sophisticated filtering and reconstruction, has emerged as an important method to dramatically speed up Monte Carlo rendering. However, the slow performance of methods like sheared filtering have limited the performance gains and interactivity. We have taken an important step towards real-time physically accurate rendering, by developing the first factored GPU 4D sheared filtering method, and expect many future developments that enable both low sampling rates and high performance.

In future work, we plan to extend our range of applications to environment lighting or spherical harmonic occlusion [30], and generalize our method for filtering Monte Carlo images involving higher dimensional integrals, including simultaneous primary and secondary distribution effects [80, 85]; initial results are shown in the appendix. A simple extension for adaptive sampling could also be considered. Furthermore, it would be thrilling to combine our reconstruction scheme with our detailed rendering methods and complex materials, so that both realism and speed can finally be achieved.

In all, we study both the complexity of the real world for realistic appearance indistinguishable from actual photographs, and the local smoothness of light transport for fast rendering. We provide a general method to synthesize details in the micron level, a physically-based way to understand appearance from microstructures, and a practical filtering method with rigorous theorecial analysis. However, even with all our previous efforts, photorealistic rendering is still far from solved. It is more and more demonstrated that both extremes have to be satisfied in order to bring people believable virtual contents: it requires even deeper studies of our current rich visual, mathematical and physical world, but this will probably break many of the assumptions that we make to trade quality for performance. Thus, even more interesting and open problems await us. We sincerely look forward to the day when Computer Generated Imagery (CGI) finally overcomes these difficulties, and opens up a gate for people to a new world.

Appendix A

Solving Equation 3.23 Analytically

To compute Equation 3.23, we need to find the integral I:

$$I = \int_{\mathbb{R}^2} G_p(\mathbf{u}) G_i(\mathbf{u}, \mathbf{s}) \, \mathrm{d}\mathbf{u}.$$

The following three steps are required: 1) Turn the 4D Gaussian $G_i(\mathbf{u}, \mathbf{s})$ into a 2D Gaussian by fixing $\mathbf{s}$. 2) Analytically compute the product of the resulting 2D Gaussians. 3) Analytically integrate the final 2D Gaussian. Below we describe these three steps in more detail.

2D slice of a 4D Gaussian. Without loss of generality, we define this problem as rewriting $g = G(\mathbf{x}; c, \mathbf{0}, \mathbf{\Sigma})$ into $g = G(\mathbf{u}; c', \mathbf{u}_0, \Sigma')$, where $\mathbf{x} = (\mathbf{u}, \mathbf{s})$ is a 4D column vector, and $\mathbf{s}$ is a 2D column vector that is fixed. Note that, here we explicitly write a multivariate Gaussian $G(\mathbf{x}; c, \mu, \Sigma)$ with its scaling coefficient c, center μ and covariance matrix Σ. (Note: in practice it is usually better to store inverse covariance matrices.)

Since $\mathbf{\Sigma}$ is symmetric, $\mathbf{\Sigma}^{-1}$ is also symmetric. We first represent $\mathbf{\Sigma}^{-1}$ using 2×2 blocks

$$\mathbf{\Sigma}^{-1} = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}, \tag{A.1}$$

Then the 4D Gaussian can be written as

$$g = c \cdot \exp\left(-\frac{1}{2}\left(\mathbf{u}^T A \mathbf{u} + 2\mathbf{s}^T B^T \mathbf{u} + \mathbf{s}^T C \mathbf{s}\right)\right). \tag{A.2}$$

Our goal is to write g as a 2D Gaussian in $\mathbf{u}$:

$$g = c' \cdot \exp\left(-\frac{1}{2}(\mathbf{u} - \mathbf{u}_0)^T \Sigma'^{-1} (\mathbf{u} - \mathbf{u}_0)\right). \tag{A.3}$$

Expanding Equation A.3 and comparing terms inside the exponential with Equation A.2, we immediately have

$$\Sigma'^{-1} = A, \;\; \mathbf{u}_0 = -A^{-1} B \mathbf{s}, \;\; c' = c \cdot \exp\left(-\frac{1}{2}(\mathbf{s}^T C \mathbf{s} - \mathbf{u}_0^T A \mathbf{u}_0)\right). \tag{A.4}$$

Product of two multivariate Gaussians. Given two multivariate Gaussians $G(\mathbf{x}; c_1, \mu_1, \Sigma_1)$ and $G(\mathbf{x}; c_2, \mu_2, \Sigma_2)$, their product is another multivariate Gaussian $G(\mathbf{x}; c, \mu, \Sigma)$, where

$$\Sigma^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1}, \qquad \mu = \Sigma(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2). \tag{A.5}$$

To find the scaling coefficient c, we just evaluate the original product at the new mean:

$$c = G(\mu; c_1, \mu_1, \Sigma_1) \cdot G(\mu; c_2, \mu_2, \Sigma_2). \tag{A.6}$$

Integral of a multivariate Gaussian. Integrating a multivariate Gaussian over $\mathbb{R}^n$ results in

$$\int_{\mathbb{R}^n} G(\mathbf{x}; c, \mu, \Sigma)\, \mathrm{d}\mathbf{x} = c \cdot (2\pi)^{n/2} |\Sigma|^{1/2}. \tag{A.7}$$

Appendix B

Alternate Wave Optics Method Derivation

Here we briefly sketch an alternative, but mathematically equivalent, derivation of our method. One can also view our method as approximating the rough surface by a set of small planar elements, or flakes, with one per grid cell. These flakes have soft overlapping boundaries in $\bar{\mathcal{S}}$, defined by a Gaussian flake shape function $K(\boldsymbol{s}) = l_k^2 G_{2D}(\boldsymbol{s}; 0, \sigma_k)$, which assuming a uniform grid, is the same for all flakes. K is normalized so that its integral is equal to the grid cell area. The flakes are centered at the grid cell centers $\boldsymbol{m}_k$.

Approximating the coherence kernel as constant over a flake with $w_k = w(\boldsymbol{m}_k - \boldsymbol{x}_\mathrm{c})$, and evaluating Equation 2.8 as a sum over the flake approximation of the surface gives:

$$f_\mathrm{r}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) \approx \frac{\xi_1}{A_\mathrm{c}} \left| \sum_k w_k \int_{\bar{\mathcal{S}}_\mathrm{c}} K(\boldsymbol{s} - \boldsymbol{m}_k) R(\boldsymbol{s})\, e^{-i\frac{2\pi}{\lambda}(\overline{\boldsymbol{\psi}} \cdot \boldsymbol{s})}\, d\boldsymbol{s} \right|^2 \tag{B.1}$$

Then we can expand $R(\boldsymbol{s})$ using Equation 2.6, substitute the per-flake planar height approximation, $H(\boldsymbol{s}) = H(\boldsymbol{m}_k) + \boldsymbol{H}'(\boldsymbol{m}_k) \cdot (\boldsymbol{s} - \boldsymbol{m}_k)$, and after some rearranging of terms, recognize that the integral has the form of a Fourier transform (Equation 2.10) of K. Combining these steps we get:

$$f_\mathrm{r}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = \frac{\xi_1}{A_\mathrm{c}} \left| \sum_k w_k \xi_2\, e^{-i2\pi\phi_k}\, \widetilde{K}\left(\frac{\xi_3 \boldsymbol{H}'(\boldsymbol{m}_k) + \overline{\boldsymbol{\psi}}}{\lambda} \right) \right|^2 \tag{B.2}$$

$$\phi_k = \frac{\xi_3 H(\boldsymbol{m}_k) + (\overline{\boldsymbol{\psi}} \cdot \boldsymbol{m}_k)}{\lambda} \tag{B.3}$$

$$\widetilde{K}(\boldsymbol{v}) = l_k^2 e^{-2\pi^2 \sigma_k^2 \|\boldsymbol{v}\|^2} \tag{B.4}$$

where we have assumed ξ_2 and ξ_3 are constant per flake.

Since $\widetilde{K}$ is a Gaussian, the expression in the sum can be interpreted as evaluations of Gabor kernels. If we expand out the expression inside the sum from our prior derivation

(Equation 3.37), we find that they match exactly. This alternate derivation may provide some additional intuition about our approximation and can also be applied to other types of flake shape functions.

Appendix C

Precomputing and Using Fur Scattering Profiles

Precomputation and Compression: We enumerate scattering coefficient $\sigma_{m,s} \in [0, 20]$ and anisotropy $g \in [0.0, 0.8]$, and we vary an additional parameter specifying different incoming directions. For azimuthal profiles, it is the offset $h' \in [-1, 1]$ assuming all sub-paths are entering the medulla horizontally (Fig. 4.13). For longitudinal profiles, it is the incident angle $\theta_i' \in [-\pi/2, \pi/2]$. We discretize the range of g into 16 steps, while for all other parameters, we use 64 steps.

We separately simulate the medulla's azimuthal and longitudinal scattering profiles using volumetric path tracing in 2D, assuming that the incident path carries unit energy. We trace a smooth unit circle azimuthally and a double slab with distance 2 in between longitudinally. For all scattering events, we use the planar Henyey-Greenstein phase function [17]

$$\rho(\theta, g) = \left(\frac{1}{2\pi}\right) \frac{1 - g^2}{1 + g^2 - 2g\cos\theta} \tag{C.1}$$

to consider anisotropy, where θ is the angle each scattering event deviates from its earlier path.

Each precomputed outgoing profile is stored using 720 bins covering every direction, recording the exiting energy. For longitudinal scattering, we further normalize the upper and lower lobes respectively, making both of them PDFs. Then we accumulate their CDFs for convenience in the next steps. Since these profiles are generally smooth, we further compress each profile by dividing it into 4 segments, and fit each segment with a quadratic function.

Refining azimuthal scattered lobes: Due to Fresnel effects, the scattered lobe could be reflected back by the surface of the medulla and the cuticle, thus undergoing further scattering, making it even smoother. We approximate this proportion as $\lambda = 1 - (1 - F(\pi/2))(1 - F(\pi/2, l))$, which is simply the leftover energy after perpendicularly transmitting through two interfaces, ignoring multiple internal reflections. So the azimuthal scattered

lobes can be extended to

$$N_p^s(h,\phi) = A_p^s(h) \cdot \left[D_p^s(h,\phi) \cdot (1-\lambda) + I_j^s \cdot \lambda \right], \tag{C.2}$$

where $I_j^s = [1 - \exp(-\sigma_{m,s} \cdot |p_j p_{j+1}|]/2\pi$ is uniform distribution of the scattered energy, under the assumption that the reflected back scattered lobe will become isotropic after more scattering events. j is the location of the vertex where the path enters the medulla and scatters, and A^s is the effective attenuation for the scattered lobe.

Normalizing longitudinal scattered lobe: Since our final longitudinal scattered lobe M^s is normalized, we have $\int M^s \,\mathrm{d}\theta_r = 1$. According to equation 4.17, we have $\int F_t \cdot C^M \,\mathrm{d}\theta_r = 1/\mu$. Then, approximately we have $\int F_t \,\mathrm{d}\theta_r \cdot \int C^M \,\mathrm{d}\theta_r = 1/\mu$. Denoting these two separated integrals as U and V, we found that U can be numerically calculated when a certain species of fur is loaded in, which introduces no overhead when rendering on the fly. For V, we first convert the integration domain into θ_r', using $\frac{\mathrm{d}\theta_r'}{\mathrm{d}\theta_r} \approx \frac{\cos\theta_r}{\eta_c \cos\theta_r'}$ by differentiating the formula for Snell's law at P' and Q'. Then we know the range of θ_r' that C^M should be integrated in. Since the precomputed C^M is a normalized PDF, the integration is a query on its CDF, which is simply accumulated in the precomputation step. Thus, we have $\mu = (UV)^{-1}$ as our normalization factor.

Appendix D

Solving Equations 4.30 and 4.31 Analytically

Equations. 4.30 and 4.31 are piecewise integrations of a polynomial and the exponential of a polynomial. The key to solving them analytically is to integrate the forms $Q_1 \cdot \exp(Q_2)$ and $C \cdot \exp(L)$, where Q_1 and Q_2 are quadratic polynomials, C is cubic, and L is linear. For simplicity, here we present the analytic result of both forms as indefinite integrations.

Integrating $\boldsymbol{Q_1 \cdot \exp(Q_2)}$:

$$\begin{aligned}&\int (dx^2 + ex + f) \cdot \exp(-ax^2 + bx + c)\,\mathrm{d}x \\ =&\frac{\sqrt{\pi}\exp\left(\frac{b^2}{4a} + c\right)\operatorname{erf}\left(\frac{2ax-b}{2\sqrt{a}}\right)(4a^2f + 2abe + 2ad + b^2d)}{8a^{5/2}} - \\ &\exp(-ax^2 + bx + c)(2adx + 2ae + bd)/(4a^2) + K\end{aligned}$$

where erf is the error function, which can be approximated with high precision using polynomials. K is the integration constant in the indefinite integral.

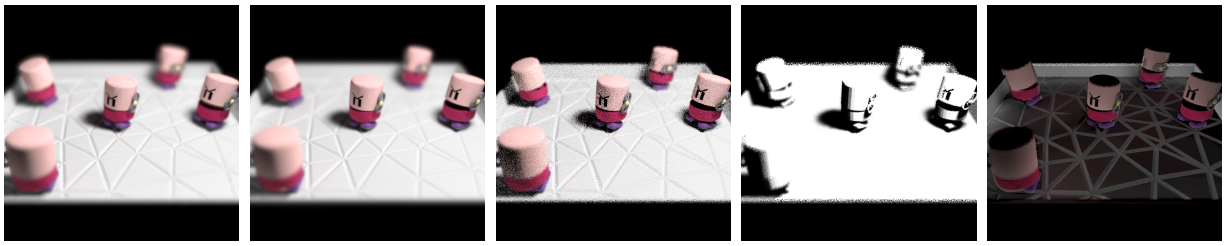
Integrating $\boldsymbol{C \cdot \exp(L)}$:

$$\begin{aligned}&\int (cx^3 + dx^2 + ex + f) \cdot \exp(-ax + b)\,\mathrm{d}x \\ =& -\exp(-ax + b)\Big(a^3cx^3 + (a^3d + 3a^2c)x^2 \\ &+ (a^3e + 2a^2d + 6ac)x + a^3f + a^2e + 2ad + 6c\Big)/a^4 + K\end{aligned}$$

where K is a constant.

Appendix E

Separating and Filtering Multiple Effects



(a) Our method, 9 spp, 580 ms **(b)** Ground truth, 12288 spp **(c)** Unfiltered **(d)** Filtered visibility **(e)** Filtered indirect illumination

Figure E.1: The TOASTERS scene rendered with an area light, depth of field and global illumination. Our method achieves visually plausible results but uses only 9 samples per pixel, each sample with 1 ray for depth of field, 1 ray for soft shadows and 1 ray for indirect illumination.

Here we focus on how to efficiently separate multiple effects (soft shadows, depth of field and diffuse global illumination) into independent single effects. Similar to [80], we observe that soft shadows and diffuse global illumination represent direct and indirect lighting respectively, and they could therefore be naturally separated. Thus, without loss of generality, we demonstrate the derivation of our separation scheme only for the combination of soft shadows and depth of field effects. Figure E.1(e) shows a separate pass for the filtered indirect illumination (using the algorithm in the main text), that is added to the final result in Fig. E.1(a).

We refer to a simplified notation, using x as (2D) screen coordinate, u as lens coordinate, and y as light coordinate. Intuitively, the pixel radiance due to direct illumination is a two-step integral. The first or inner step filters out the correct outgoing radiance due to the area

light, and the second or outer step filters for the lens. The equation is given by

$$L_{dir}(x) = \int_u \left(\int_y f(x,u,y)V(x,u,y)\,\mathrm{d}y \right) k(x,u)\,\mathrm{d}u. \tag{E.1}$$

where $f(x,u,y)$ is the BRDF term, $V(x,u,y)$ is the visibility term, and $k(x,u)$ represents the texture or reflectance. Note that, this equation is difficult to separate, because both the BRDF term and the visibility term depend on samples from the lens and samples from the area light.

To solve the problem, we first denote the product of the BRDF term and the visibility term as $F(x,u,y) = f(x,u,y) \cdot V(x,u,y)$. Then we introduce an approximation by replacing F for each pixel with its average over every related lens sample u, or $F(x,u,y) \approx \bar{F}(x,y)|_u$. Then equation E.1 becomes

$$L_{dir}(x) \approx \int_u \left(\int_y \bar{F}(x,y)|_u\,\mathrm{d}y \right) k(x,u)\,\mathrm{d}u. \tag{E.2}$$

We have essentially factored out the inner integral over the light for soft shadows, and the outer integral over the lens for defocus.

We now propose a two-step filtering algorithm. For each pixel, we sample the lens to shoot primary rays. For each valid hit, we shoot one (or more) secondary shadow rays and compute the corresponding F term. After this, we average primary rays to get $\bar{F}$.

The following filtering steps are straightforward. We first filter the $\bar{F}(x,y)$ light field samples, using our proposed four-step fast sheared filtering. Then we filter the resulting (x,u) light field from the previous step according to the lens to get the depth of field effect, again with our fast sheared filtering algorithm. Figure E.1 shows the final result as well as different stages using our method. Compared to the claimed running time of 3.61 seconds for the same scene using axis-aligned filtering in [80], we achieve a 6× speed up, yet achieving a visually convincing result, although differences are noticeable as compared to the ground truth.

Note that, the approximation we proposed is almost as conservative as that introduced in [80]. In practice, it also guarantees accuracy. When the effect of u range is small, the approximated $\bar{F}$ is accurate. This indicates that the closer to the focal plane, the more accurate the approximation is. For those areas far from the focal plane, since the outer integral (lens filter) dominates, the output image is largely blurred, so minor inaccuracies of the visibility approximation could be neglected.

The time complexity of our separation scheme is still $O(nl)$, because we simply perform our fast sheared filtering twice. For the storage, the approximation allows us to store the samples for each effect separately, i.e. the storage cost is still 4D rather than 6D. So we consider it practical and efficient for our algorithm to handle multiple effects.

Bibliography

[1] Sameer Agarwal, Keir Mierle, et al. *Ceres Solver*. http://ceres-solver.org. 2010.

[2] T Annen et al. "Real-time all-frequency shadows in dynamic scenes". In: *ACM Transactions on Graphics (Proc. SIGGRAPH 08)* 27.3 (2008), Article 34, 1–8.

[3] U. Assarsson and T. Möller. "A Geometry-Based Soft Shadow Volume Algorithm Using Graphics Hardware". In: *ACM Transactions on Graphics (SIGGRAPH 03)* 22.3 (2003), pp. 511–520.

[4] P. Beckmann and A. Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Artech House Radar Library. Books on Demand, 1968. ISBN: 9780835742313. URL: http://books.google.com/books?id=nn92AAAACAAJ.

[5] Laurent Belcour and Pascal Barla. "A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence". In: *ACM Trans. Graph.* 36.4 (July 2017), 65:1–65:14. ISSN: 0730-0301. DOI: 10.1145/3072959.3073620. URL: http://doi.acm.org/10.1145/3072959.3073620.

[6] Laurent Belcour et al. "5D Covariance tracing for efficient defocus and motion blur". In: *ACM Transanctions on Graphics* 32.3 (2013), 31:1–31:18.

[7] Brent Burley. "Physically-Based Shading at Disney". In: *Technical Report* (2012).

[8] Ellen Carrlee and Lauren Horelick. "The Alaska Fur ID Project: A virtual resource for material identification". In: *Objects Specialty Group Postprints*. Vol. 18. American Institute for Conservation of Historic and Artistic Works, 2011, pp. 149–171.

[9] Jin-Xiang Chai et al. "Plenoptic Sampling". In: *Proceedings of SIGGRAPH 00*. 2000, pp. 307–318.

[10] Matt Jen-Yuan Chiang et al. "A Practical and Controllable Hair and Fur Model for Production Path Tracing". In: *Computer Graphics Forum* 35.2 (2016), pp. 275–283. ISSN: 1467-8659. DOI: 10.1111/cgf.12830. URL: http://dx.doi.org/10.1111/cgf.12830.

[11] Petrik Clarberg and Jacob Munkberg. "Deep Shading Buffers on Commodity GPUs". In: *ACM Trans. Graph.* 33.6 (Nov. 2014), 227:1–227:12. ISSN: 0730-0301. DOI: 10.1145/2661229.2661245. URL: http://doi.acm.org/10.1145/2661229.2661245.

[12] Robert L. Cook, Loren Carpenter, and Edwin Catmull. "The REYES Image Rendering Architecture". In: SIGGRAPH '87. 1987, pp. 95–102.

[13] C. Crassin et al. "Interative indirect illumination using voxel cone tracing". In: *Computer Graphics Forum* 30.7 (2011), pp. 1921–1930.

[14] F. Crow. "Shadow Algorithms for Computer Graphics". In: *SIGGRAPH 77*. 1977, pp. 242–248.

[15] Tom Cuypers et al. "Reflectance Model for Diffraction". In: *ACM Trans. Graph.* 31.5 (2012), 122:1–122:11.

[16] Holger Dammertz et al. "Edge-avoiding À-Trous Wavelet Transform for Fast Global Illumination Filtering". In: *Proceedings of the Conference on High Performance Graphics*. Saarbrucken, Germany, 2010, pp. 67–75.

[17] AnthonyB. Davis. "Effective Propagation Kernels in Structured Media with Broad Spatial Correlations, Illustration with Large-Scale Transport of Solar Photons Through Cloudy Atmospheres". English. In: *Computational Methods in Transport*. Ed. by Frank Graziani. Vol. 48. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2006, pp. 85–140. ISBN: 978-3-540-28122-1. DOI: 10.1007/3-540-28125-8_5. URL: http://dx.doi.org/10.1007/3-540-28125-8_5.

[18] Douglas W Deedrick and Sandra L Koch. "Microscopy of Hair Part 1: A Practical Guide and Manual for Human Hairs". In: *Forensics Science Communication* (2004).

[19] Douglas W Deedrick and Sandra L Koch. "Microscopy of hair part II: a practical guide and manual for animal hairs". In: *Forensics Science Communication* (2004).

[20] Mauricio Delbracio et al. "Boosting Monte Carlo Rendering by Ray Histogram Fusion". In: *ACM Transactions on Graphics* 33.1 (2014), 8:1–8:15. ISSN: 0730-0301. DOI: 10.1145/2532708. URL: http://doi.acm.org/10.1145/2532708.

[21] Eugene d'Eon, Steve Marschner, and Johannes Hanika. "A Fiber Scattering Model with Non-separable Lobes". In: *ACM SIGGRAPH 2014 Talks*. Vancouver, Canada, 2014, 46:1–46:1. ISBN: 978-1-4503-2960-6. DOI: 10.1145/2614106.2614161. URL: http://doi.acm.org/10.1145/2614106.2614161.

[22] Eugene d'Eon, Steve Marschner, and Johannes Hanika. "Importance Sampling for Physically-based Hair Fiber Models". In: *SIGGRAPH Asia 2013 Technical Briefs*. Hong Kong, Hong Kong, 2013, 25:1–25:4. ISBN: 978-1-4503-2629-2. DOI: 10.1145/2542355.2542386. URL: http://doi.acm.org/10.1145/2542355.2542386.

[23] Eugene d'Eon et al. "An Energy-conserving Hair Reflectance Model". In: *ACM Transactions on Graphics (TOG)*. 2011, pp. 1181–1187. DOI: 10.1111/j.1467-8659.2011.01976.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2011.01976.x.

[24] Daljit Singh Dhillon et al. "Interactive Diffraction from Biological Nanostructures". In: *Computer Graphics Forum* 33.8 (2014), pp. 177–188. ISSN: 1467-8659. DOI: 10.1111/cgf.12425. URL: http://dx.doi.org/10.1111/cgf.12425.

[25] Zhao Dong et al. "Predicting Appearance from Measured Microgeometry of Metal Surfaces". In: *ACM Trans. Graph.* 35.1 (2015). ISSN: 0730-0301. DOI: 10.1145/2815618. URL: http://doi.acm.org/10.1145/2815618.

[26] Craig Donner et al. "An Empirical BSSRDF Model". In: *ACM Trans. Graph.* 28.3 (2009), 30:1–30:10. ISSN: 0730-0301. DOI: 10.1145/1531326.1531336. URL: http://doi.acm.org/10.1145/1531326.1531336.

[27] Daniel Dunbar and Greg Humphreys. "A Spatial Data Structure for Fast Poisson-disk Sample Generation". In: *ACM Trans. Graph.* 25.3 (2006). ISSN: 0730-0301. DOI: 10.1145/1141911.1141915. URL: http://doi.acm.org/10.1145/1141911.1141915.

[28] Jonathan Dupuy et al. "Linear Efficient Antialiased Displacement and Reflectance Mapping". In: *ACM Trans. Graph.* 32.6 (2013). URL: http://hal.inria.fr/hal-00858220.

[29] F Durand et al. "A Frequency Analysis of Light Transport". In: *ACM Transactions on Graphics* 24.3 (2005), pp. 1115–1126.

[30] K. Egan, F. Durand, and R. Ramamoorthi. "Practical Filtering for Efficient Ray-Traced Directional Occlusion". In: *ACM Transactions on Graphics (SIGGRAPH Asia 11)* 30.6 (2011a).

[31] K Egan et al. "Frequency Analysis and Sheared Filtering for Shadow Light Fields of Complex Occluders". In: *ACM Transactions on Graphics* 30.2 (2011b), 9:1–9:13.

[32] K Egan et al. "Frequency analysis and sheared reconstruction for rendering motion blur". In: *ACM Transactions on Graphics* 28.3 (2009), 93:1–93:13.

[33] Eric Enderton et al. "Stochastic transparency". In: *IEEE transactions on visualization and computer graphics* 17.8 (2011), pp. 1036–1047.

[34] Antonín Galatík et al. *Furskin Identification*. http://www.furskin.cz. 2011.

[35] Alan Genz. "Numerical Computation of Rectangular Bivariate and Trivariate Normal and T Probabilities". In: *Statistics and Computing* 14.3 (2004), pp. 251–260.

[36] Dan B. Goldman. "Fake Fur Rendering". In: *SIGGRAPH 97*. 1997, pp. 127–134. ISBN: 0-89791-896-7. DOI: 10.1145/258734.258807. URL: http://dx.doi.org/10.1145/258734.258807.

[37] G. Guennebaud, L. Barthe, and M. Paulin. "High-Quality Adaptive Soft Shadow Mapping". In: *Computer Graphics Forum* 26.3 (2007), pp. 525–533.

[38] G. Guennebaud, L. Barthe, and M. Paulin. "Real-time Soft Shadow Mapping by Backprojection". In: *EGSR 06*. 2006, pp. 227–234.

[39] T Hachisuka et al. "Multidimensional Adaptive Sampling and Reconstruction for Ray Tracing". In: *ACM Transactions on Graphics* 27.3 (2008), 33:1–33:10.

[40] Charles Han et al. "Frequency Domain Normal Map Filtering". In: *ACM Trans. Graph.* 26.3 (2007).

[41] James Harvey. "Fourier treatment of near-field scalar diffraction theory". In: *American Journal of Physics* 47.11 (1979), pp. 974–980. DOI: 10.1119/1.11600.

[42] James E. Harvey and Richard N. Pfisterer. "Evolution of the transfer function characterization of surface scatter phenomena". In: *Proc.SPIE* 9961 (2016), DOI: 10.1117/12.2237083. URL: http://dx.doi.org/10.1117/12.2237083.

[43] J. Hasenfratz et al. "A survey of Real-Time Soft Shadow Algorithms". In: *Computer Graphics Forum* 22.4 (2003), pp. 753–774.

[44] Ken Hashimoto. "The structure of human hair". In: *Clinics in Dermatology* 6.4 (1988), pp. 7–21. ISSN: 0738-081X. DOI: http://dx.doi.org/10.1016/0738-081X(88)90060-0. URL: http://www.sciencedirect.com/science/article/pii/0738081X88900600.

[45] Xiao D. He et al. "A Comprehensive Physical Model for Light Reflection". In: *SIGGRAPH Comput. Graph.* 25.4 (1991), pp. 175–186. ISSN: 0097-8930. DOI: 10.1145/127719.122738. URL: http://doi.acm.org/10.1145/127719.122738.

[46] Eric Heitz et al. "The SGGX Microflake Distribution". In: *ACM Trans. Graph.* 34.4 (July 2015), 48:1–48:11. ISSN: 0730-0301. DOI: 10.1145/2766988. URL: http://doi.acm.org/10.1145/2766988.

[47] Christophe Hery and Ravi Ramamoorthi. "Importance sampling of reflection from hair fibers". In: *Journal of Computer Graphics Techniques (JCGT)* 1.1 (2012), pp. 1–17.

[48] Nicolas Holzschuch and Romain Pacanowski. "A Two-scale Microfacet Reflectance Model Combining Reflection and Diffraction". In: *ACM Trans. Graph.* 36.4 (2017), 66:1–66:12. ISSN: 0730-0301. DOI: 10.1145/3072959.3073621. URL: http://doi.acm.org/10.1145/3072959.3073621.

[49] Homan Igehy. "Tracing Ray Differentials". In: SIGGRAPH 1999. 1999.

[50] Homan Igehy. "Tracing Ray Differentials". In: *SIGGRAPH 99*. 1999, pp. 179–186.

[51] Wenzel Jakob. *Mitsuba renderer*. http://www.mitsuba-renderer.org. 2010.

[52] Wenzel Jakob and Steve Marschner. "Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport". In: *ACM Trans. Graph.* 31.4 (2012), 58:1–58:13.

[53] Wenzel Jakob, Christian Regg, and Wojciech Jarosz. "Progressive Expectation–Maximization for Hierarchical Volumetric Photon Mapping". In: *Computer Graphics Forum (Proceedings of EGSR 2011)* 30.4 (2011).

[54] Wenzel Jakob et al. "A Radiative Transfer Framework for Rendering Materials with Anisotropic Structure". In: *ACM Trans. Graph.* 29.4 (July 2010), 53:1–53:13.

[55] Wenzel Jakob et al. "Discrete Stochastic Microfacet Models". In: *ACM Trans. Graph.* 33.4 (2014).

[56] Wenzel Jakob et al. "Discrete Stochastic Microfacet Models". In: *ACM Trans. Graph.* 33.4 (2014).

[57] Henrik Wann Jensen and Juan Buhler. "A rapid hierarchical rendering technique for translucent materials". In: *ACM Transactions on Graphics (TOG)*. Vol. 21. 3. ACM. 2002, pp. 576–581.

[58] Henrik Wann Jensen et al. "A practical model for subsurface light transport". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 511–518.

[59] G. Johnson et al. "Soft irregular shadow mapping: fast, high-quality, and robust soft shadows". In: *I3D 2009*. 2009, pp. 57–66.

[60] J Kajiya. "The Rendering Equation". In: *Proceedings of SIGGRAPH 86*. 1986, pp. 143–150.

[61] J. Kajiya and T. Kay. "Rendering Fur with Three Dimensional Textures". In: *SIGGRAPH 89*. 1989, pp. 271–280.

[62] Nima Khademi Kalantari and Pradeep Sen. "Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms". In: *Computer Graphics Forum (Proc. of Eurographics 2013)* 32.2 (2013), pp. 93–102.

[63] Pramook Khungurn and Steve Marschner. "Azimuthal Scattering from Elliptical Hair Fibers". In: *To appear in ACM Transactions on Graphics* (2015).

[64] Andrey Krywonos. "Predicting Surface Scatter Using A Linear Systems Formulation Of Non-paraxial Scalar Diffraction". PhD thesis. University of Central Florida, 2006.

[65] S. Laine et al. "Soft shadow volumes for ray tracing". In: *ACM Transactions on Graphics* 24.3 (Aug. 2005), pp. 1156–1165.

[66] Ann M Lanari et al. "Wave optics simulation of statistically rough surface scatter". In: *Earth Observing Systems XXII*. Vol. 10402. International Society for Optics and Photonics. 2017, p. 1040215.

[67] Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. "Real-time Lens Blur Effects and Focus Control". In: *ACM Trans. Graph.* 29.4 (July 2010), 65:1–65:7.

[68] Jaakko Lehtinen et al. "Reconstructing the Indirect Light Field for Global Illumination". In: *ACM Transanctions on Graphics* 31.4 (2012), 51:1–51:10.

[69] Jaakko Lehtinen et al. "Temporal Light Field Reconstruction for Rendering Distribution Effects". In: *ACM Transactions on Graphics* 30.4 (2011), 55:1–55:12.

[70] Kefei Lei and John F. Hughes. "Approximate Depth of Field Effects Using Few Samples Per Pixel". In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '13. ACM, 2013, pp. 119–128. DOI: 10.1145/2448196.2448215.

[71] Anat Levin et al. "Fabricating BRDFs at High Spatial Resolution Using Wave Optics". In: *ACM Trans. Graph.* 32.4 (2013), 144:1–144:14.

[72] Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. "SURE-based Optimization for Adaptive Sampling and Reconstruction". In: *ACM Transactions on Graphics* 31.6 (2012), 186:1–186:9.

[73] Tom Lokovic and Eric Veach. "Deep Shadow Maps". In: *SIGGRAPH 00*. 2000, pp. 385–392.

[74] L. Mandel and E. Wolf. *Optical Coherence and Quantum Optics*. Cambridge University Press, 1995. ISBN: 9780521417112. URL: http://books.google.com/books?id=FeBix14iM70C.

[75] Stephen R. Marschner et al. "Light Scattering from Human Hair Fibers". In: *ACM Transactions on Graphics (TOG)* 22.3 (2003), pp. 780–791. ISSN: 0730-0301. DOI: 10.1145/882262.882345. URL: http://doi.acm.org/10.1145/882262.882345.

[76] Heylal Mashaal et al. "First direct measurement of the spatial coherence of sunlight". In: *Opt. Lett.* 37.17 (2012), pp. 3516–3518. DOI: 10.1364/OL.37.003516. URL: http://ol.osa.org/abstract.cfm?URI=ol-37-17-3516.

[77] M. McCool. "Anisotropic diffusion for Monte Carlo noise reduction". In: *ACM Transactions on Graphics* 18.2 (1999), pp. 171–194.

[78] S. Mehta, B. Wang, and R. Ramamoorthi. "Axis-Aligned Filtering for Interactive Sampled Soft Shadows". In: *ACM Transactions on Graphics* 31.6 (2012), 163:1–163:10.

[79] Soham Uday Mehta et al. "Axis-Aligned Filtering for Interactive Physically-Based Diffuse Indirect Lighting". In: *ACM Transactions on Graphics* 32.4 (2013), 96:1–96:12. URL: http://graphics.berkeley.edu/papers/Udaymehta-IPB-2013-07/.

[80] S. Mehta et al. "Factored Axis-Aligned Filtering for Rendering Multiple Distribution Effects". In: *ACM Transactions on Graphics* 33.5 (2014).

[81] D Mitchell. "Spectrally Optimal Sampling for Distribution Ray Tracing". In: *SIGGRAPH 91*. 1991, pp. 157–164.

[82] Don Mitchell and Pat Hanrahan. "Illumination from Curved Reflectors". In: *SIGGRAPH Comput. Graph.* 26.2 (1992), pp. 283–291.

[83] Jonathan T. Moon and Stephen R. Marschner. "Simulating Multiple Scattering in Hair Using a Photon Mapping Approach". In: *ACM Trans. Graph.* 25.3 (2006), pp. 1067–1074. ISSN: 0730-0301. DOI: 10.1145/1141911.1141995. URL: http://doi.acm.org/10.1145/1141911.1141995.

[84] Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner. "Rendering discrete random media using precomputed scattering solutions". In: EGSR 07. 2007, pp. 231–242.

[85] Jacob Munkberg et al. "Layered Reconstruction for Defocus and Motion Blur". In: *Computer Graphics Forum*. Vol. 33. 4. Wiley Online Library. 2014, pp. 81–92.

[86] Hubert Nguyen and William Donnelly. "Hair animation and rendering in the nalu demo". In: *GPU Gems* 2 (2005), pp. 361–380.

[87] Shinji Ogaki, Yusuke Tokuyoshi, and Sebastian Schoellhammer. "An empirical fur shader". In: *ACM SIGGRAPH ASIA 2010 Sketches*. ACM. 2010, p. 16.

[88] J.A. Ogilvy. *Theory of wave scattering from random rough surfaces*. A. Hilger, 1991. ISBN: 9780750300636.

[89] Marc Olano and Dan Baker. "LEAN Mapping". In: I3D 2010. ACM, 2010, pp. 181–188.

[90] Jiawei Ou et al. "ISHair: Importance Sampling for Hair Scattering". In: *ACM SIGGRAPH 2012 Talks*. Los Angeles, California, 2012, 28:1–28:1. ISBN: 978-1-4503-1683-5. DOI: 10.1145/2343045.2343084. URL: http://doi.acm.org/10.1145/2343045.2343084.

[91] R Overbeck, C Donner, and R Ramamoorthi. "Adaptive Wavelet Rendering". In: *ACM Transactions on Graphics* 28.5 (2009).

[92] S Paris and F Durand. "A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach". In: *9th European Conference on Computer Vision (ECCV 2006)*. May 2006, pp. 568–580.

[93] Pieter Peers et al. "A Compact Factored Representation of Heterogeneous Subsurface Scattering". In: *ACM Trans. Graph.* 25.3 (2006), pp. 746–753. ISSN: 0730-0301. DOI: 10.1145/1141911.1141950. URL: http://doi.acm.org/10.1145/1141911.1141950.

[94] Georg Petschnigg et al. "Digital photography with flash and no-flash image pairs". In: *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*. New York, NY, USA: ACM, 2004, pp. 664–672. DOI: http://doi.acm.org/10.1145/1186562.1015777.

[95] PolyCub. *PolyCub: Cubature over Polygonal Domains*. http://cran.r-project.org/web/packages/polyCub/. Accessed: 2014-01-14. 2004.

[96] Michael Potmesil and Indranil Chakravarty. "A lens and aperture camera model for synthetic image generation". In: *Proceedings of SIGGRAPH 81*. 1981, pp. 297–305.

[97] R Ramamoorthi and P Hanrahan. "A Signal-Processing Framework for Inverse Rendering". In: *Proceedings of SIGGRAPH 01*. 2001, pp. 117–128.

[98] R. Ramamoorthi et al. "A Theory of Monte Carlo Visibility Sampling". In: *ACM Transactions on Graphics* 31.5 (2012).

[99] Peiran Ren et al. "Global Illumination with Radiance Regression Functions". In: *ACM Trans. Graph.* 32.4 (2013), 130:1–130:12. ISSN: 0730-0301. DOI: 10.1145/2461912.2462009. URL: http://doi.acm.org/10.1145/2461912.2462009.

[100] Zhong Ren et al. "Interactive Hair Rendering Under Environment Lighting". In: *ACM SIGGRAPH 2010 Papers*. SIGGRAPH '10. Los Angeles, California: ACM, 2010, 55:1–55:8. ISBN: 978-1-4503-0210-4. DOI: 10.1145/1833349.1778792. URL: http://doi.acm.org/10.1145/1833349.1778792.

[101] Tobias Ritschel et al. "Micro-Rendering for Scalable, Parallel Final Gathering". In: *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2009)* 28.5 (2009).

[102] T. Ritschel et al. "The State of the Art in Interactive Global Illumination". In: *Computer Graphics Forum* 31.1 (2012), pp. 160–188.

[103] F. Rouselle, C. Knaus, and M. Zwicker. "Adaptive Rendering with Non-Local Means Filtering". In: *ACM Transactions on Graphics* 31.6 (2012), 195:1–195:11.

[104] Martin Rump et al. "Photo-realistic Rendering of Metallic Car Paint from Image-Based Measurements". In: *Computer Graphics Forum* 27.2 (2008), pp. 527–536.

[105] H. Rushmeier and G. Ward. "Energy preserving non-linear filters". In: (1994), pp. 131–138.

[106] Iman Sadeghi et al. "A Practical Microcylinder Appearance Model for Cloth Rendering". In: *ACM Trans. Graph.* 32.2 (2013), 14:1–14:12. ISSN: 0730-0301. DOI: 10.1145/2451236.2451240. URL: http://doi.acm.org/10.1145/2451236.2451240.

[107] Iman Sadeghi et al. "An Artist Friendly Hair Shading System". In: *ACM Trans. Graph.* 29.4 (2010), 56:1–56:10. ISSN: 0730-0301. DOI: 10.1145/1778765.1778793. URL: http://doi.acm.org/10.1145/1778765.1778793.

[108] P. Sen and S. Darabi. "On Filtering the Noise from the Random Parameters in Monte Carlo Rendering". In: *ACM Transactions on Graphics* 31.3 (2012), 18:1–18:15.

[109] P. Shirley et al. "A local image reconstruction algorithm for stochastic rendering". In: *ACM Symposium on Interactive 3D Graphics*. 2011, pp. 9–14.

[110] Erik Sintorn and Ulf Assarsson. "Hair Self Shadowing and Transparency Depth Ordering Using Occupancy Maps". In: *Symposium on Interactive 3D Graphics and Games*. Boston, Massachusetts, 2009, pp. 67–74. ISBN: 978-1-60558-429-4. DOI: 10.1145/1507149.1507160. URL: http://doi.acm.org/10.1145/1507149.1507160.

[111] Erik Sintorn and Ulf Assarsson. "Real-time approximate sorting for self shadowing and transparency in hair rendering". In: *Proceedings of the 2008 symposium on Interactive 3D graphics and games*. ACM. 2008, pp. 157–162.

[112] P Sloan, J Kautz, and J Snyder. "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments". In: *ACM Transactions on Graphics (Proc. SIGGRAPH 02)* 21.3 (2002), pp. 527–536.

[113] Peter-Pike Sloan, Jan Kautz, and John Snyder. "Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments". In: *ACM Trans. Graph.* 21.3 (2002), pp. 527–536. ISSN: 0730-0301. DOI: 10.1145/566654.566612. URL: http://doi.acm.org/10.1145/566654.566612.

[114] C Soler and F Sillion. "Fast Calculation of Soft Shadow Textures Using Convolution". In: *SIGGRAPH 98*. 1998, pp. 321–332.

[115] C Soler et al. "Fourier depth of field". In: *ACM Transactions on Graphics* 28.2 (2009), 18:1–18:12.

[116] Jos Stam. "Diffraction Shaders". In: *SIGGRAPH 99*. New York, NY, USA, 1999, pp. 101–110. ISBN: 0-201-48560-5. DOI: 10.1145/311535.311546. URL: http://dx.doi.org/10.1145/311535.311546.

[117] Jos Stam. "Multiple scattering as a diffusion process". English. In: *Rendering Techniques '95*. 1995, pp. 41–50. ISBN: 978-3-211-82733-8. DOI: 10.1007/978-3-7091-9430-0_5. URL: http://dx.doi.org/10.1007/978-3-7091-9430-0_5.

[118] Robert F Stamm, Mario L Garcia, and Judith J Fuchs. "The optical properties of human hair I. Fundamental considerations and goniophotometer curves". In: *Journal of the Society of Cosmetic Chemists* 28.9 (1977), pp. 571–599.

[119] George G Stokes. "On the intensity of the light reflected from or transmitted through a pile of plates". In: *Proceedings of the Royal Society of London* 11 (1860), pp. 545–556.

[120] Jerry Tessendorf. "Simulating ocean water". In: *Technical Report* (1999).

[121] Antoine Toisoul and Abhijeet Ghosh. "Practical Acquisition and Rendering of Diffraction Effects in Surface Reflectance". In: *ACM Trans. Graph.* 36.5 (July 2017), 166:1–166:16. ISSN: 0730-0301. DOI: 10.1145/3012001. URL: http://doi.acm.org/10.1145/3012001.

[122] M. Toksvig. "Mipmapping normal maps". In: *Journal of Graphics Tools* 10.3 (2005), pp. 65–71.

[123] K Torrance and E Sparrow. "Theory for Off-Specular Reflection from Roughened Surfaces". In: *JOSA* 57.9 (1967), pp. 1105–1114.

[124] Yuting Tsai and Zenchung Shih. "All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation". In: *ACM Transactions on Graphics (TOG)* 25.3 (2006), pp. 967–976.

[125] Chi-Wei Tseng. "A Physically-Based Reflectance Model For Mammalian Fur Fibers Based On Anatomy And Gonioreflectometry Measurements". MA thesis. University of California San Diego, 2015.

[126] Karthik Vaidyanathan et al. "Layered Light Field Reconstruction for Defocus Blur". In: *ACM Transactions on Graphics* 34.2 (2015), 23:1–23:12.

[127] M Alex O Vasilescu and Demetri Terzopoulos. "TensorTextures: multilinear image-based rendering". In: *ACM Transactions on Graphics (TOG)* 23.3 (2004), pp. 336–342.

[128] E Veach. "Robust Monte Carlo Methods for Light Transport Simulation". PhD thesis. Stanford University, 1997.

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

[130] Zdravko Velinov, Sebastian Werner, and Matthias B. Hullin. "Real-Time Rendering of Wave-Optical Effects on Scratched Surfaces". In: *Computer Graphics Forum 37 (2) (Proc. EUROGRAPHICS)* 37.2 (2018).

[131] Jakob Verbeek, Jan Nunnink, and Nikos Vlassis. "Accelerated EM-based clustering of large data sets". In: *Data Mining and Knowledge Discovery* 13.3 (2006), pp. 291–307.

[132] Bruce Walter et al. "Microfacet Models for Refraction Through Rough Surfaces". In: EGSR 07. 2007, pp. 195–206.

[133] Bruce Walter et al. "Single Scattering in Refractive Media with Triangle Mesh Boundaries". In: *ACM Trans. Graph.* 28.3 (2009), 92:1–92:8.

[134] H. Wang et al. "Out-of-core tensor approximation of multi-dimensional matrices of visual data". In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 527–535.

[135] Rui Wang, John Tran, and David Luebke. "All-frequency Interactive Relighting of Translucent Objects with Single and Multiple Scattering". In: *ACM Trans. Graph.* 24.3 (2005), pp. 1202–1207. ISSN: 0730-0301. DOI: 10.1145/1073204.1073333. URL: http://doi.acm.org/10.1145/1073204.1073333.

[136] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. ISSN: 1057-7149. DOI: 10.1109/TIP.2003.819861.

[137] Xue Wei. *What is human hair? A light and scanning electron microscopy study.* 2006. URL: http://www.optics.rochester.edu/workgroups/cml/opt307/spr06/xue/project.htm.

[138] Sebastian Werner et al. "Scratch Iridescence: Wave-optical Rendering of Diffractive Surface Structure". In: *ACM Trans. Graph.* 36.6 (2017), 207:1–207:14. ISSN: 0730-0301. DOI: 10.1145/3130800.3130840. URL: http://doi.acm.org/10.1145/3130800.3130840.

[139] Kun Xu et al. "A Practical Algorithm for Rendering Interreflections with All-frequency BRDFs". In: *ACM Trans. Graph.* 33.1 (2014), 10:1–10:16.

[140] Kun Xu et al. "Interactive Hair Rendering and Appearance Editing under Environment Lighting". In: *ACM Transactions on Graphics* 30.6 (2011), 173:1–173:10.

[141] Ling-Qi Yan, Henrik Wann Jensen, and Ravi Ramamoorthi. "An Efficient and Practical Near and Far Field Fur Reflectance Model". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36.4 (2017).

[142] Ling-Qi Yan et al. "A BSSRDF Model for Efficient Rendering of Fur with Global Illumination". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2017)* 36.6 (2017).

[143] Ling-Qi Yan et al. "Fast 4D Sheared Filtering for Interactive Rendering of Distribution Effects". In: *ACM Transactions on Graphics* (2015).

[144] Ling-Qi Yan et al. "Physically-Accurate Fur Reflectance: Modeling, Measurement and Rendering". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2015)* (2015).

[145] Ling-Qi Yan et al. "Position-Normal Distributions for Efficient Rendering of Specular Microstructure". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2016)* 35.4 (2016).

[146] Ling-Qi Yan et al. "Position-normal Distributions for Efficient Rendering of Specular Microstructure". In: *ACM Trans. Graph.* 35.4 (2016), 56:1–56:9. ISSN: 0730-0301. DOI: 10.1145/2897824.2925915. URL: http://doi.acm.org/10.1145/2897824.2925915.

[147] Ling-Qi Yan et al. "Rendering Glints on High-resolution Normal-mapped Specular Surfaces". In: *ACM Trans. Graph.* 33.4 (2014). ISSN: 0730-0301. DOI: 10.1145/2601097.2601155. URL: http://doi.acm.org/10.1145/2601097.2601155.

[148] Ling-Qi Yan et al. "Rendering Specular Microgeometry with Wave Optics". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018)* 37.4 (2018).

[149] Xuan Yu, Rui Wang, and Jingyi Yu. "Real-time Depth of Field Rendering via Dynamic Light Field Generation and Filtering". In: *Computer Graphics Forum* 29.7 (2010), pp. 2099–2107. ISSN: 1467-8659.

[150] Xuan Yu et al. "A Framework for Rendering Complex Scattering Effects on Hair". In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '12. Costa Mesa, California: ACM, 2012, pp. 111–118. ISBN: 978-1-4503-1194-6. DOI: 10.1145/2159616.2159635. URL: http://doi.acm.org/10.1145/2159616.2159635.

[151] Cem Yuksel and John Keyser. "Deep Opacity Maps". In: *Computer Graphics Forum* 27.2 (2008), pp. 675–680. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2008.01165.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2008.01165.x.

[152] Arno Zinke and Andreas Weber. "Light Scattering from Filaments". In: *IEEE Transactions on Visualization and Computer Graphics* 13.2 (2007), pp. 342–356. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.43. URL: http://dx.doi.org/10.1109/TVCG.2007.43.

[153] Arno Zinke et al. "A Practical Approach for Photometric Acquisition of Hair Color". In: *ACM Transactions on Graphics* 28.5 (2009), 165:1–165:9. ISSN: 0730-0301. DOI: 10.1145/1618452.1618511. URL: http://doi.acm.org/10.1145/1618452.1618511.

[154] Arno Zinke et al. "Dual Scattering Approximation for Fast Multiple Scattering in Hair". In: *ACM Transactions on Graphics (TOG)* 27.3 (2008), 32:1–32:10. ISSN: 0730-0301. DOI: 10.1145/1360612.1360631. URL: http://doi.acm.org/10.1145/1360612.1360631.

[155] Tobias Zirr and Anton S Kaplanyan. "Real-time rendering of procedural multiscale materials". In: I3D 2016. ACM, 2016, pp. 139–148.