MODELING AND RENDERING FABRICS AT MICRON-RESOLUTION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Shuang Zhao

August 2014

MODELING AND RENDERING FABRICS AT MICRON-RESOLUTION

Shuang Zhao, Ph.D.

Cornell University 2014

Fabrics are essential to our everyday lives. Recently advances in physically based rendering techniques and computing power have made it possible to accurately model and reproduce their appearance. This has led to many applications in not only computer graphics but also other fields including industrial design, retail, and entertainment.

Traditionally, fabrics are treated as infinitely thin 2D sheets. These surface-based reflectance models, although being conceptually simple, have insufficient power to describe a fabric's small-scale 3D geometries, such as disorganized layers of fibers in felts. Thus, these models cannot accurately reproduce the fabric's thickness and fuzziness, limiting the level of realism they can offer.

In contrast, volumetric models, which has recently been very successful in modeling fabric appearance, explicitly express those fiber-level structures, upon which greatly varying visual effects, from anisotropic highlights to deep textures, emerge automatically. However, volumetric models are generally difficult to build: one would need to specify spatially varying optical properties at high (e.g., micron) resolutions.

This dissertation presents a family of algorithms that introduce a brand new way to automatically build and efficiently render volumetric fabric models at micron-resolution. These models capture rich details at fiber-level, yielding highly realistic renderings even under extreme close-ups.

Our first contribution tackles the challenge of automated creation of high

fidelity fabric models. Our method uses volume imaging techniques to measure a fabric's detailed structural information. Such information is then combined with a photograph to form a complete model through an appearance matching process. The resulting model offers fabric renderings with unprecedented quality.

CT scanning fabric samples can be highly time consuming. Our second contribution aims for creating volumetric models for woven fabrics with complex designs under minimal measurement cost. Our approach starts with building a small database of cloth samples with elementary weave patterns. Then, given an input pattern, a volumetric synthesis stage is performed to form the final volume by copying contents from the database while keeping visible artifacts (such as seams) minimized. Fabric models created by this method have been used by textile researchers at Rhode Island School of Design to preview the appearances of their designs.

Our third work focuses on efficient rendering of high-resolution fabric volumes. Based on the observation that these volumes contain repeated structures (i.e., multiple instances of the same content in the database), we precompute light transport information for those structures, and a single precomputation can be reused for many designs synthesized from the same database. During the rendering process, the precomputed information is modularly combined through a Monte Carlo Matrix Inversion (MCMI) framework. This method has accelerated the rendering of thick fabrics by an order of magnitude.

Finally, we switch gears and introduce high-order similarity relations to computer graphics. This theory originates in applied physics and studies when two sets of material scattering parameters would lead to identical appearance. We introduce a numerical algorithm to utilize this theory in its general high-order

form. The practical usefulness of our method is demonstrated using forward and inverse rendering of translucent media.

The approaches presented in this dissertation have created fabric renderings with unprecedented fidelity. Remaining challenges include developing more general synthesis technique to support a wider range of fabric structures (such as knitworks) as well as finding more powerful light transport models and inverse rendering algorithms so that the rendered fabrics match photographs under a wide combination of viewing and lighting configurations. We also believe the techniques introduced in this dissertation can provide valuable insights for developing appearance modeling techniques for general material beyond fabrics.

BIOGRAPHICAL SKETCH

Shuang Zhao was born December 8th, 1984 in Wuhan, China. After graduating from Shanghai Jiao Tong University in 2007 with a Bachelor of Engineering in computer science, he spent one year at Microsoft Research Asia as a visiting scholar. Since 2008, Shuang has been studying for his doctorate degree in computer science at Cornell University.

To the memory of my grandparents, Kai Jiang and Jie Wu.

ACKNOWLEDGMENTS

First and foremost, I am deeply grateful to my advisor Kavita Bala for her vision, inspiration and superb guidance. During the years at Cornell, she has taught me not only techniques for solving specific research problems, but also how to think as a computer scientist. Without her continuous support, this thesis would not have been possible.

I would like to thank Steve Marschner for his help on many pieces of this dissertation. His insightful advices have been invaluable. I wish to express my gratitude to Bruce Walter, who has kindly answered my numerous rendering-related questions.

I am grateful to Robert Kleinberg for his inspiring lectures on algorithm design and analysis, to Charles Van Loan for teaching me finding and exploiting structures in matrices and tensors, to Shane Henderson for showing me mathematical and statistical foundations of stochastic process and Monte Carlo simulation, and to Doug James for presenting me physical laws governing how objects move and make sound as well as practical algorithms for simulating these effects.

I also thank the members of my special committee: Kavita Bala, Robert Kleinberg, Charles Van Loan, and Steve Marschner. Their constructive suggestions have significantly improved my work contributing to this thesis.

I feel extremely fortunate to have the chance working with many excellent collaborators on research projects related and beyond this thesis: Kavita Bala, Steve Marschner, Ravi Ramamoorthi, Todd Zickler, Edward Adelson, Ramesh Raskar, Bruce Walter, Wenzel Jakob, Miloš Hašan, Ioannis Gkioulekas, Bei Xiao, Nikhil Naik, Andreas Velten, and Edgar Velázquez-Armendáriz. Without their

efforts, going through the SIGGRAPH and SIGGRAPH Asia deadlines would have never been so smooth.

My thanks to my office and lab mates, who have created a stimulating environment and enriched my experience at Cornell: Adam Arbree, Miloš Hašan, Changxi Zheng, Edgar Velázquez-Armendáriz, Wenzel Jakob, Daniel Hauagge, Sean Bell, Pramook Khungurn, Daniel Schroeder, and Timothy Langlois.

Finally, I thank my family for their unfaltering love and support. I thank my wife Siman for the great deal of love and understanding. I thank my parents Bin and Tao: it was them who introduced me to computers and BASIC/LOGO programming when I was a child! I thank my grandparents Kai and Jie for caring and teaching me, for picking me from school, and for the new-year gifts. Sadly, they are no longer here to watch me completing this thesis. I dedicate this thesis to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Fabrics are important to our lives: they are used to create many everyday essentials including clothing as well as functional cloth such as curtains, tablecloth, and bedsheets (Figure 1.1). Acquiring, modeling, and computationally reproducing the appearance of fabrics, therefore, has been an active research area in computer graphics for decades. These techniques can lead to applications in many fields such as

- **Virtual prototyping:** textile designers can *preview* the appearance of their new designs before physically constructing them (on looms). This allows one to quickly explore many possible designs without investing time and money on building physical prototypes.
- **Retail:** with accurate predictions of fabric deformation and appearance, *virtual dressing rooms* can be built. By providing their 3D body scans, shoppers can try out clothes virtually. This is particularly useful for online shopping where physical samples are not available to shoppers.
- **Entertainment:** people can experience highly realistic fabrics in *CG movies* and *video games*.

Clothing Window Curtain Tablecloth Bedsheets

Figure 1.1: **Fabrics** in our everyday lives.

(a) (b) (c)

Figure 1.2: **Close-up photographs of three kinds of fabrics:** (a) silk satin; (b) denim; (c) velvet.

Unfortunately, accurately modeling and reproducing a fabric's appearance in the virtual world remains very challenging. First, the appearances of different kinds of fabrics vary greatly. Silk satin, for example, normally looks light weight, smooth, and shiny (Figure 1.2a); denims are much more diffuse and strongly textured (Figure 1.2b); velvet, on the other hand, appears heavier with characteristic grazing-angle highlights (Figure 1.2c). It is difficult to capture all these varying appearances with one universal model. Second, many thick fabrics, such as velvet and felt, contain complicated yet visible 3D geometries that cannot be fully described by commonly used surface-based reflectance models. Furthermore, such geometry in real fabrics usually contains naturally arising irregularities that are challenging to model analytically or procedurally but crucial to the result's realism.

In this dissertation, we focus on two major problems: the *automated creation* and *efficient rendering* of fabric appearance models.

Model Creation. Surface-based reflectance models are widely used in computer graphics. They model light-material interaction as light bounding off the material's surface. Unfortunately, these models cannot accurately capture the

detailed geometries of fabrics which contribute significantly to their appearance. We, therefore, need to model fabrics as actual 3D volumes. Recent advances in radiative transfer theory [43] and fabric appearance modeling [107] have led to sophisticated volumetric models that are capable of capturing a fabric's fiber-level details. These models have been very successful in reproducing the detailed fiber-level geometry of fabrics. However, it remains difficult to create these complicated models in an automated manner. Existing methods often require manually writing down equations or programs to describe the 3D arrangement of yarns or fibers for each type of fabric. This process requires much human effort and is time consuming. Further, the resulting models look unrealistically "perfect" because they lack visually important features like naturally occurring irregularities. In this thesis, we develop fundamentally new ways to build highly realistic volumetric models for fabrics while requiring minimal user input.

Rendering Algorithms. Although volumetric models offer great representative power, rendering them requires a massive, sometimes even impractical, amount of computation. This is because for volumetric models, light can travel inside the material and *scatter* multiple times before getting absorbed or leaving the material volume. Simulating all these subsurface scattering events can be very expensive. We introduce new algorithms to efficiently render volumetric fabric models, greatly improving their practical usefulness.

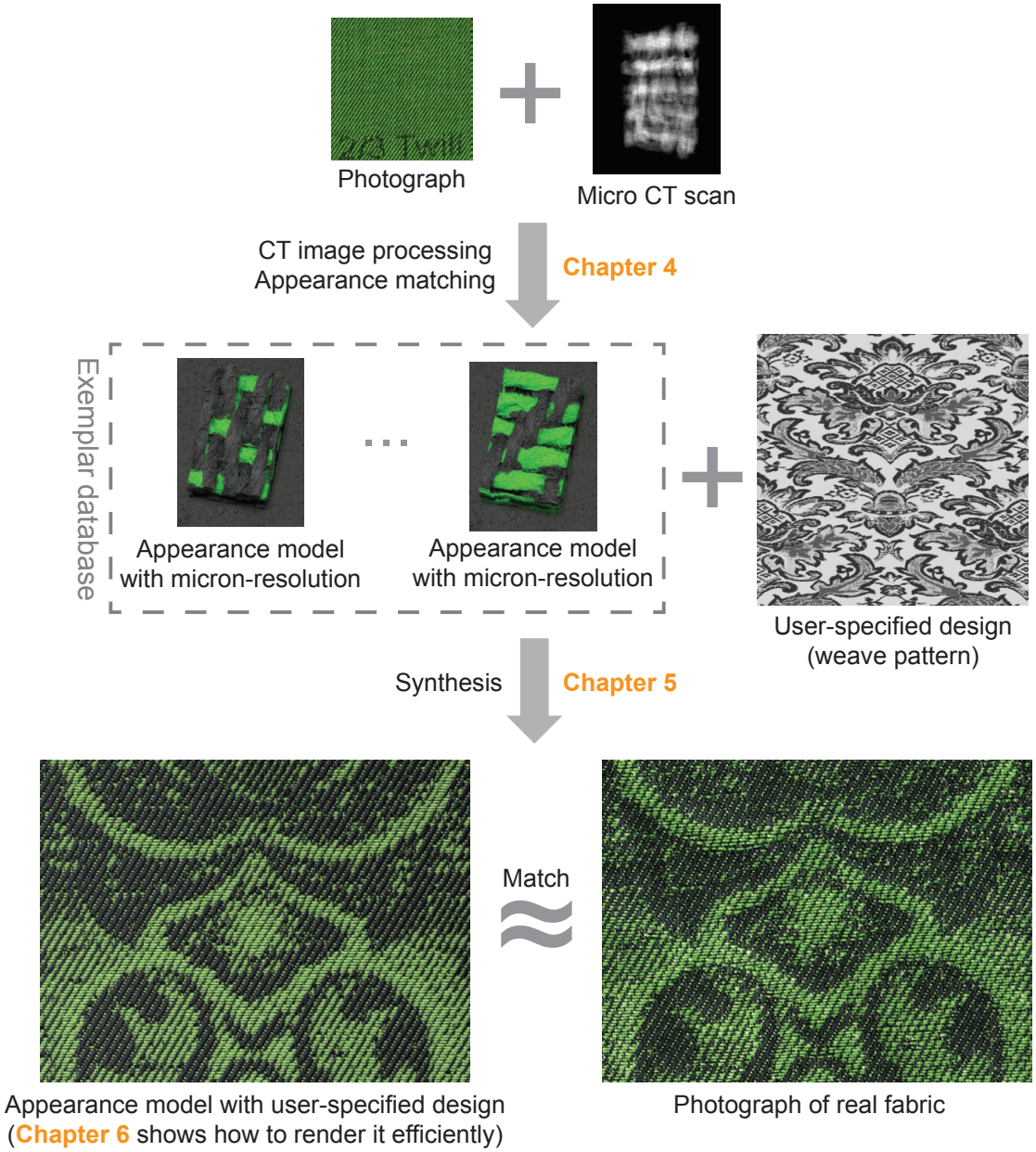
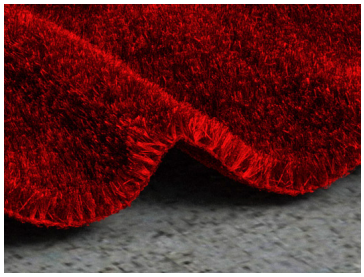


Figure 1.3: **The main pipeline of our technique** presented in this dissertation. (a) We present a brand new method (Chapter 4) that automatically builds micron-resolution fabric appearance models by combining photographs and micro computed tomography (CT) scans of real cloth samples. (b) Using fabric models with elementary patterns, our method can synthesize highly complicated models with user-specified designs (Chapter 5). (c) The renderings (Chapter 6) of our models match physically constructed samples very well, and can be used to predict the appearance of fabric designs.

1.1 Contributions and outline

In this thesis, we present a family of fundamentally new approaches to model and render fabrics. Our end-to-end pipeline (Figure 1.3) spans the full range from material appearance modeling and representation to physically based rendering. The micron-resolution models built by our pipeline can produce rendered images matching photographs of real fabric samples with very good accuracy, making our approach particularly useful for virtual prototyping of textile designs.

This section summarizes our four major technical contributions and provides an outline for the rest of this thesis.



Contribution 1. Our first contribution (Chapter 4) is a new approach for acquiring volume models, based on density data from X-ray micro CT scans[1] and appearance data from photographs under uncontrolled illumination. To model a material, a CT scan is made, yielding a scalar density volume. This 3D data has micron resolution details about the structure of cloth but lacks all optical information. So we combine this density data with a reference photograph of the cloth sample to infer its optical properties. We show that this approach can easily produce volume appearance models with extreme detail, and at larger scales the distinctive textures and highlights of a range of very different fabrics like satin and velvet emerge automatically — all based simply on having accurate mesoscale geometry.

[1]Micro CT scans can readily be ordered from a number of facilities at the cost of a couple hundred dollars per sample. With the development of CT technology, such cost has been decreasing.

Contribution 2. To date, micro CT scanners can only measure samples up to $0.5 \times 0.5\ \mathrm{cm}^2$ in size (while maintaining micron-resolution). Unfortunately, many fabrics have designs far beyond this extent. Our second contribution (Chapter 5) is a volumetric synthesis framework overcoming this obstacle. This approach starts with user-specified fabric designs and produces models that correctly capture the yarn-level structural details of cloth. We create a small database of volumetric exemplars using our previous approach. To build an output model, our method synthesizes a new volume by copying data from the exemplars to match a weave pattern that specifies the desired output structure. Our results demonstrate that our approach generalizes well to complex designs and can produce highly realistic results at both large and small scales.

Contribution 3. Fabric models synthesized by our technique have offered a new level of realism. However, rendering these highly detailed models is very expensive. Our third contribution (Chapter 6) is a precomputation based approach that accelerates the rendering process by an order of magnitude, significantly improving the practical usefulness of our fabric models. The approach precomputes light transport for all basic building blocks in an exemplar database. This single precomputation can then be reused to speed up the rendering of any volume synthesized using this database. At render time, when those building blocks are tiled to produce a high-resolution volume, we accurately compute low-order scattering, and use a novel modular flux transfer algorithm to approximate higher-order

scattering (which usually takes the majority of computation).

Contribution 4. Our fourth contribution (Chapter 7) is about radiative transfer theory, the theoretical foundation for simulating light transport within translucent media (including fabrics). We introduce similarity theory, which originates in applied physics, to computer graphics. This theory introduces a set of equivalence relations over material parameters such that those belonging to one equivalence class yield identical appearances (when the radiance field is band-limited in the angular domain). We then introduce practical algorithms to utilize this theory in its most general form. Our technique can benefit both forward and inverse rendering of translucent media.

The rest of this dissertation is organized as follows.

- **Chapter 2** overviews background on rendering and fabric structures.
- **Chapter 3** discusses prior work on fabric appearance modeling and physically based rendering of translucent materials.
- **Chapter 4** introduces our new method for using micro CT scans combined with a photograph to construct micron-resolution fabric models.
- **Chapter 5** describes our volumetric synthesis algorithm for producing complex models with user-specified designs.
- **Chapter 6** presents our precomputaton based algorithm that accelerates the rendering of our detailed fabric models.
- **Chapter 7** provides a complete exposition of similarity theory and introduces practical algorithms to utilize it with full generality.

- **Chapter 8** presents our conclusion and suggestions for future research directions.

The techniques described in this thesis were presented at multiple ACM SIGGRAPH conferences [111, 112, 110, 113] and have been used by textile researchers at Rhode Island School of Design to build a textile visualization system for better design efficiency. In the future, these methods can form a building block to create high-quality appearance models for materials beyond cloth. The challenge remains to develop underlying light transport models with sufficient generality and accuracy, as well as to define basic building blocks of general materials and to find more powerful synthesis algorithms that merge such blocks into normal-sized objects.

CHAPTER 2

BACKGROUND

This chapter reviews relevant material and introduces important terminology that will be used extensively in the rest of this dissertation.

Section 2.1 presents background on physically based rendering of scattering (translucent) media. Section 2.2 describes the geometry and creation process of cloth, which provide important insights for modeling fabric appearance. Most materials covered in this chapter is high-level. For more in-depth discussions, please refer to the related work cited in the following sections.

2.1 Rendering basics

The main goal of physically based rendering is to synthesize images that accurately represent the appearance of the objects in a fully described scene. More precisely, the rendered image captures all light received by a virtual camera's image sensor. The scene description includes a complete specification of geometries and material properties of the objects as well as lighting and camera configurations. The light may arrive at the image sensor directly, or indirectly through surface reflections/refractions and volume scatterings.

2.1.1 The physics of light transport

The physical laws governing light transport in a scene exist in a hierarchy of increasing approximations. *Quantum optics* explains the dual wave-particle nature of light and is the most accurate model up-to-date. *Geometric optics*, on the other hand, is the simplest model. This model focuses on situations where the scale

of objects is much larger than the wavelength of light, so that physical effects such as diffraction and interference are negligible. This is generally the case for computer graphics applications. Thus, the geometric optics model is used virtually exclusively in physically based rendering.

In particular, geometric optics assume the following on the behavior of light:

- Light travels in straight lines in vacuum and is not affected by external factors such as gravity fields (i.e., no "gravitational lensing").
- Light travels instantaneously through any medium.
- Light is incoherent and unpolarized.

In this dissertation, we also rely on these assumptions. Since we treat fabrics as translucent media, the rest of this section briefly reviews general theories and algorithms on rendering this type of materials.

2.1.2 Radiative transfer

Under the geometric optics model, when light travels inside a medium, it interacts with the material through *absorption* and *scattering*. These normally occur due to interactions between photons and optically active contents of the medium, such as cloth fibers for fabrics and water droplets for fog and clouds. However, a normal-sized object usually contains a massive number of such contents, making it impractical to consider each of them individually.

The *radiative transfer framework* [7] assumes the medium to be *random* and describes light propagation based on its *statistical interpretation*. In this framework, a photon travels along a straight line for a certain distance, which is determined

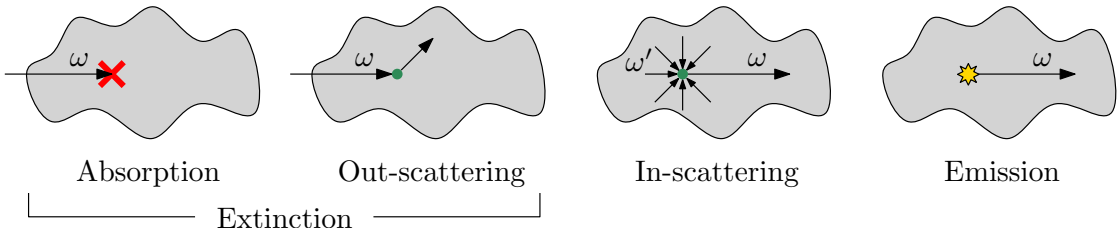


Figure 2.1: **Physical interpretations** for the different terms in the radiative transfer equation.

by the optical density of that path, and then either gets absorbed (and disappear) or scattered into a new direction. These interactions are described mathematically through the *radiative transfer equation* (RTE). This equation, in the form usually used in computer graphics, is

$$\begin{aligned}&(\boldsymbol{\omega}\cdot\nabla)L(\mathbf{x},\boldsymbol{\omega})\\&=-\sigma_t(\mathbf{x})L(\mathbf{x},\boldsymbol{\omega})+\sigma_s(\mathbf{x})\int_{\mathbb{S}^2}f(\mathbf{x},\boldsymbol{\omega}'\to\boldsymbol{\omega})L(\mathbf{x},\boldsymbol{\omega}')\,\mathrm{d}\boldsymbol{\omega}'+Q(\mathbf{x},\boldsymbol{\omega})\end{aligned}\tag{2.1}$$

which needs to hold at every location x inside the medium volume. In this equation, L is the quantity *radiance*. Given a point x and direction $\boldsymbol{\omega}$, the quantity $L(\mathbf{x},\boldsymbol{\omega})$ describes the amount of light passing through x in direction $\boldsymbol{\omega}$.[1] Other terms in (2.1) include: σ_s and σ_t, the *scattering* and *extinction* (or *attenuation*) *coefficients*; f, the *phase function*; and Q, the *volume source* term representing emitted radiance. In addition, $\sigma_a := \sigma_t - \sigma_s$ and $\alpha := \sigma_s/\sigma_t$ are usually called the *absorption coefficient* and the *single-scattering albedo*, respectively.

Figure 2.1 illustrates the physical interpretations of the three terms on the RHS of (2.1). The first of these three, $-\sigma_t(\mathbf{x})L(\mathbf{x},\boldsymbol{\omega})$, accounts for the radiance loss due to photons colliding with the medium contents (such as water droplets or cloth fibers). When such a collision happens, a photon can be either absorbed, or scattered into another direction (called *out-scattering*) and no longer contributes to $L(\mathbf{x},\boldsymbol{\omega})$.

[1] Please refer to [82, 95] for a thorough review on radiance and other related quantities.

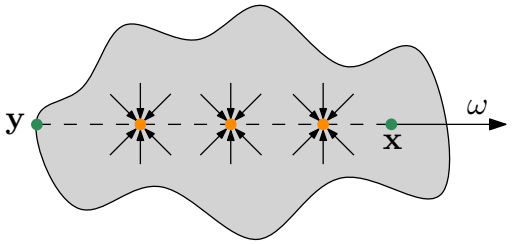


Figure 2.2: **Volume rendering equation:** evaluation of $L(\mathbf{x}, \boldsymbol{\omega})$ through integration over the dashed line segment.

The second term on the RHS of (2.1), $\sigma_s(\mathbf{x}) \int_{\mathbb{S}^2} f(\mathbf{x}, \boldsymbol{\omega}' \to \boldsymbol{\omega}) L(\mathbf{x}, \boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}'$, captures photons which travel passing $\mathbf{x}$ along other directions and then are scattered into direction $\boldsymbol{\omega}$ (called *in-scattering*). This term convolves the directional radiance and the phase function (at location $\mathbf{x}$) over the unit sphere $\mathbb{S}^2$ to obtain the total radiance that needs to be added to $L(\mathbf{x}, \boldsymbol{\omega})$. The integration is done using the solid angle measure. The phase function $f(\boldsymbol{\omega}' \to \boldsymbol{\omega})$, as a function of $\boldsymbol{\omega}$, is a probability distribution over $\mathbb{S}^2$ that expresses the chance that a photon traveling along $\boldsymbol{\omega}'$ will be redirected into $\boldsymbol{\omega}$.

The last term in (2.1), $Q(\mathbf{x}, \boldsymbol{\omega})$, specifies the amount of illumination emitted by the medium itself.

As a whole, the RTE (2.1) describes that the directional derivative of $L(\mathbf{x}, \boldsymbol{\omega})$ along $\boldsymbol{\omega}$ is determined by the negative contribution from extinction plus the positive one from in-scattering and self-emission.

To evaluate $L(\mathbf{x}, \boldsymbol{\omega})$ at some location $\mathbf{x}$ interior to a medium, we can integrate both sides of the RTE (2.1) for a line segment of length s with endpoints $\mathbf{x}$ and $\mathbf{y} := \mathbf{x} - s\boldsymbol{\omega}$ (as shown in Figure 2.2), yielding the *volume rendering equation*:

$$L(\mathbf{x}, \boldsymbol{\omega}) = \int_{\mathbf{y}}^{\mathbf{x}} \tau(\mathbf{x}', \mathbf{x}) \left(Q(\mathbf{x}', \boldsymbol{\omega}) + \sigma_s(\mathbf{x}') \int_{\mathbb{S}^2} f(\mathbf{x}', \boldsymbol{\omega}' \to \boldsymbol{\omega}) L(\mathbf{x}', \boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}' \right) \mathrm{d}\mathbf{x}' + \tau(\mathbf{y}, \mathbf{x}) L(\mathbf{y}, \boldsymbol{\omega}) \tag{2.2}$$

where

$$\tau(\mathbf{u}, \mathbf{v}) := \exp\left(-\int_{\mathbf{u}}^{\mathbf{v}} \sigma_t(\mathbf{x}')\,\mathrm{d}\mathbf{x}'\right)$$

captures the *transmittance* between $\mathbf{u}$ and $\mathbf{v}$.

2.1.3 Monte Carlo integration

Due to its complexity, general analytic solutions to the volume rendering equation (2.2) do not exist. In physically based rendering, Monte Carlo methods are usually used to compute (2.2) numerically.

The basic idea of Monte Carlo integration is fairly simple. Consider a real-valued function $g(X)$. We would like to compute the integral of g over some domain Ω:

$$I := \int_{\Omega} g(t)\,\mathrm{d}t.$$

Let X to be a continuous random variable over Ω with probability density p_X. Assume p_X to be positive everywhere g is non-zero. Then, it is easy to verify that

$$\mathbb{E}_X\left[\frac{g(t)}{p_X(t)}\right] = I.$$

Let $x_1, x_2, \ldots, x_N$ be N independent realizations of X and

$$\langle I \rangle := \frac{1}{N}\sum_{i=1}^{N} \frac{g(x_i)}{p_X(x_i)}.$$

The strong law of large numbers guarantees that $\langle I \rangle$ converges to I almost surely when $N \to \infty$. Thus, $\langle I \rangle$ is called an *estimator* of I.

Given the function g, the choice of p_X significantly affects the convergence rate of the estimator. Assume g is nonnegative, a perfect p_X is proportional to g (assuming g is nonnegative):

$$p_X(x) = \frac{g(x)}{\int_{\Omega} g(t)\,\mathrm{d}t}. \tag{2.3}$$

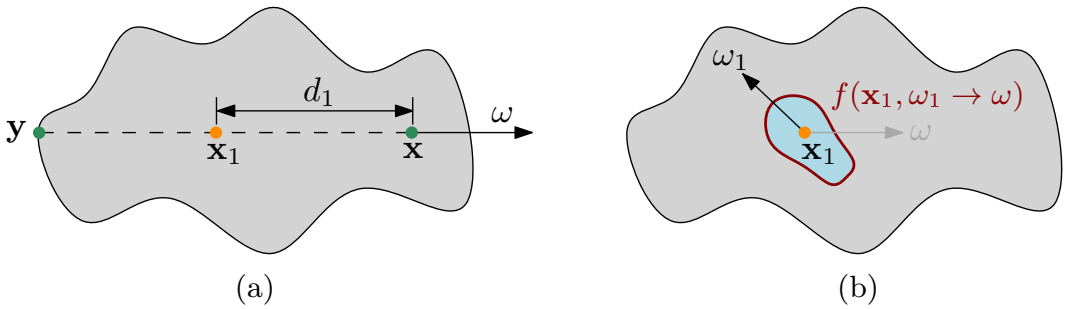


Figure 2.3: **Volume path tracing.** Given a location $\mathbf{x}$ and a direction $\boldsymbol{\omega}$: (a) new position $\mathbf{x}_1$ is first sampled along $(\mathbf{x}, -\boldsymbol{\omega})$; (b) new direction $\boldsymbol{\omega}_1$ is drawn according to the phase function f at $\mathbf{x}_1$.

In this case, the estimator will have zero variance. But (2.3) requires knowing I in the first place. In practice, a good sampling distribution should match the shape of g as closely as possible.

A main advantage of Monte Carlo integration is its conceptual simplicity: it does not require the function g to have any simple analytical form. Comparing to other (deterministic) numerical integration methods, such as Simpson's rule, the Monte Carlo method converges more slowly (at a rate of $N^{-1/2}$) but scales to high dimensional problems more easily.

2.1.4 Volume path tracing

Monte Carlo integration can be used to evaluate (2.2) through *volume path tracing*. In this subsection, we describe a unidirectional version of this method.

Assume that the radiance is known everywhere except for the interior of the medium volume. In order to evaluate $L(\mathbf{x}, \boldsymbol{\omega})$ for some given location $\mathbf{x}$ (within a medium) and direction $\boldsymbol{\omega}$, one needs to first compute the double integral in

Algorithm 2.1 Pseudocode for the volume path tracing algorithm.

function COMPUTERADIANCE($\mathbf{x}, \boldsymbol{\omega}$)
  Find $\mathbf{y}$ by tracing a ray from $\mathbf{x}$ in direction $-\boldsymbol{\omega}$
  Draw $d_1 \in \mathbb{R}_+$ from probability density (2.6)
  if $d_1 < \|\mathbf{x} - \mathbf{y}\|_2$ **then**
    $\mathbf{x}_1 \leftarrow \mathbf{x} - d_1\boldsymbol{\omega}$
    Sample $\boldsymbol{\omega}_1 \in \mathbb{S}^2$ according to (2.7)
    return $(Q(\mathbf{x}_1, \boldsymbol{\omega}) + \sigma_s(\mathbf{x}_1) \cdot \text{COMPUTERADIANCE}(\mathbf{x}_1, \boldsymbol{\omega}_1))/\sigma_t(\mathbf{x}_1)$
  else
    return $L(\mathbf{y}, \boldsymbol{\omega})$
  end if
end function

(2.2). Using Monte Carlo integration, this boils down to evaluating

$$\mathbb{E}_{d_1}\left[\frac{\tau(\mathbf{x}_1, \mathbf{x})}{p_1(d_1)}\left(Q(\mathbf{x}_1, \boldsymbol{\omega}) + \sigma_s(\mathbf{x}_1)\int_{\mathbb{S}^2} f(\mathbf{x}_1, \boldsymbol{\omega}' \to \boldsymbol{\omega}) L(\mathbf{x}_1, \boldsymbol{\omega}')\,\mathrm{d}\boldsymbol{\omega}'\right)\right] \tag{2.4}$$

where $\mathbf{x}_1 := \mathbf{x} - d_1\boldsymbol{\omega}$ and d_1 is randomly sampled from probability density p_1 (Figure 2.3-a). Since (2.4) contains another integral, an extra Monte Carlo step needs to be applied, yielding

$$\mathbb{E}_{d_1}\left[\frac{\tau(\mathbf{x}_1, \mathbf{x})}{p_1(d_1)}\left(Q(\mathbf{x}_1, \boldsymbol{\omega}) + \sigma_s(\mathbf{x}_1)\,\mathbb{E}_{\boldsymbol{\omega}_1}\left[\frac{f(\mathbf{x}_1, \boldsymbol{\omega}_1 \to \boldsymbol{\omega})}{q_1(\boldsymbol{\omega}_1; \mathbf{x}_1, \boldsymbol{\omega})} L(\mathbf{x}_1, \boldsymbol{\omega}_1)\right]\right)\right] \tag{2.5}$$

where $\boldsymbol{\omega}_1$ is sampled from q_1 given $\mathbf{x}_1$ and $\boldsymbol{\omega}$ (Figure 2.3-b). Finally, $L(\mathbf{x}_1, \boldsymbol{\omega}_1)$ is computed recursively.

To implement (2.5), one needs to pick probability density functions p_1 and q_1. In practice, it is common to use

$$p_1(d) = \sigma_t(\mathbf{x} - d\boldsymbol{\omega})\,\tau(\mathbf{x}, \mathbf{x} - d\boldsymbol{\omega}), \tag{2.6}$$

$$q_1(\boldsymbol{\omega}'; \mathbf{x}_1, \boldsymbol{\omega}) = f(\mathbf{x}_1, \boldsymbol{\omega}' \to \boldsymbol{\omega}) \tag{2.7}$$

which leads to significant simplification of (2.5):

$$\mathbb{E}_{d_1}\left[\frac{Q(\mathbf{x}_1, \boldsymbol{\omega}) + \sigma_s(\mathbf{x}_1)\,\mathbb{E}_{\boldsymbol{\omega}_1}[L(\mathbf{x}_1, \boldsymbol{\omega}_1)]}{\sigma_t(\mathbf{x}_1)}\right]. \tag{2.8}$$

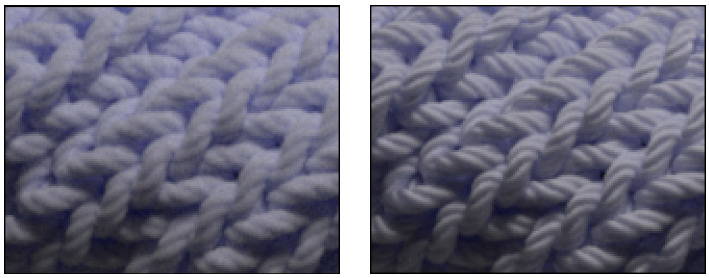
Figure 2.4: **Renderings** of (from left to right): soaps, olive oil, blue curacao, and reduced fat milk created using volume path tracing [30].

Given (2.6), it is easy to verify that the probability for d_1 to be greater than the distance between $\mathbf{x}$ and $\mathbf{y}$, say s, exactly equals $\tau(\mathbf{x}, \mathbf{y})$. Thus, to account for the second term on the RHS of (2.2), the algorithm only needs to return $L(\mathbf{y}, \boldsymbol{\omega})$ when $d_1 \geq s$.

The entire flow of this algorithm is summarized in Algorithm 2.1. To implement line 3 for heterogeneous media, techniques such as Woodcock tracking [103, 109] can be applied.

During the recursive execution of Algorithm 2.1, a *light path* $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, \ldots$ is constructed. The average contribution of N such paths is an unbiased estimator of $L(\mathbf{x}, \boldsymbol{\omega})$.

Multiple techniques can be used to improve the convergence rate of the volume path tracing algorithm. *Next event estimation*, for example, splits the in-scattered radiance into *single-scattered* and *multiple-scattered* components and evaluates the former by sampling the light source and the latter by sampling the phase function. *Multiple importance sampling* (MIS) combines several sampling



Isotropic Anisotropic

Figure 2.5: **Zoomed renderings of a scarf** represented with isotropic (left) and anisotropic (right) media [43]. Accounting for anisotropy leads to more realistic highlights and color variations.

methods to further improve the performance of the estimator.

The main strength of path tracing is its unbiasedness: by increasing N, the estimated value always converges to the correct solution. For a detailed and more complete description on path tracing and its many variants with more sophisticated sampling strategies, such as bidirectional path tracing (BDPT) and Metropolis light transport (MLT), please see [95, 41].

In this dissertation, we use volume path tracing (with MSI) to generate all the rendered images in Chapters 4, 5, 6 and 7.

2.1.5 Anisotropic media

Recently, Jakob et al. [43] extended the radiative transfer framework described in Section 2.1.2 to better handle *anisotropic media* containing oriented non-spherical content, such as fabrics (see Figure 2.5). In this model, the absorption, scattering and extinction coefficients become functions of both location $\mathbf{x}$ and direction $\boldsymbol{\omega}$, and the RTE (2.1) is generalized to an anisotropic version:

$$(\boldsymbol{\omega} \cdot \nabla) L(\boldsymbol{\omega}) = -\sigma_t(\boldsymbol{\omega}) L(\boldsymbol{\omega}) + \sigma_s(\boldsymbol{\omega}) \int_{\mathbb{S}^2} f(\boldsymbol{\omega}' \to \boldsymbol{\omega}) L(\boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}' + Q(\boldsymbol{\omega}) \qquad (2.9)$$

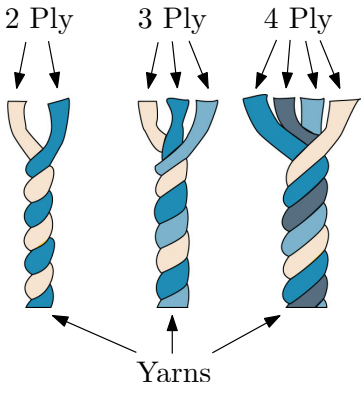
where the spatial dependencies are dropped for better readability. Fortunately, the volume path tracing algorithm described in Section 2.1.4 generalizes naturally to solve (2.9).

In this thesis, we model fabrics as anisotropic media. To provide the data required by this powerful formulation, we present a piece of fabric as a 3D volume in which $\sigma_t(\mathbf{x}, \boldsymbol{\omega})$, $\sigma_s(\mathbf{x}, \boldsymbol{\omega})$, and $f(\mathbf{x}, \boldsymbol{\omega}' \to \boldsymbol{\omega})$ are specified for all locations $\mathbf{x}$ and all directions $\boldsymbol{\omega}, \boldsymbol{\omega}'$. Note that, to build such a volumetric model at fiber resolution, the parameters have to be provided at billions or even trillions of densely sampled locations. Automated creation of these models, therefore, is necessary yet nontrivial. In Chapters 4 and 5, we introduce an end-to-end pipeline that builds volumetric fabric models at micron-resolution.

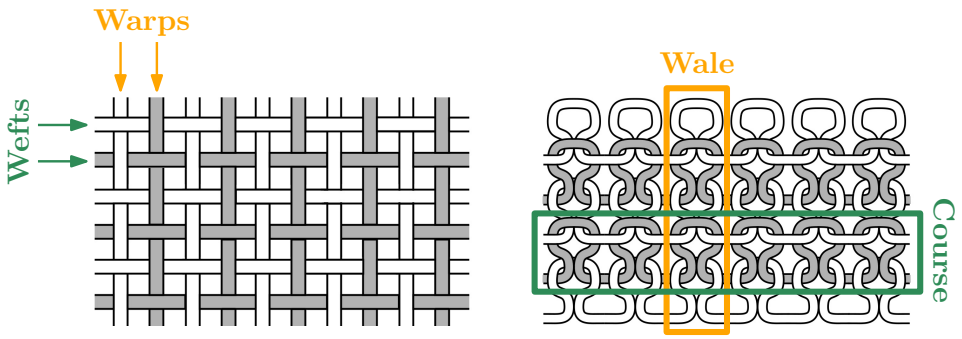
2.2 Fabrics as a glance

Understanding the underlying geometry of real fabrics can provide useful insights for modeling their appearance. The discussion in this section serve as a general introduction to fabrics and weaving, and is drawn from [51].

Fabrics are composed of *yarns*, which themselves are produced by twisting natural (e.g., silk, cotton) or artificial (e.g., polyester) *fibers* to form long strands. Yarns can also contain multiple *plies*, where each ply is a single group of twisted fibers (see the figure to the right). All these factors have significant impact on the appearance of a yarn.



Fabrics can generally be divided into three categories: *felt*, *knitted*, and *woven*.



(a) Woven (b) Knitted

Figure 2.6: Yarn structures of a woven and a knitting pattern [51].

Felts are built by pressing together individual fibers, forming a disorganized and structurally stable layer of fibers. Knitted and woven fabrics, on the other hand, contain regular and repeated patterns.

Woven fabrics (which are the main focus in Section 5) are formed by interlacing two sets of yarns, called *warps* and *wefts*. During the weaving process, warp yarns are spun onto a loom, and weft yarns are inserted from a perpendicular direction. At each crossing of a warp and a weft yarn, one of them goes on top of the other. The specification of this configuration is called the *weave pattern* (see Figure 2.6-a). Although weaving may seem simple in principle, depending on the mechanical properties of the yarns and the loom, fabrics with greatly varying structures and appearances can be created.

Knitted fabric contains a regular set of loops called *stitches*. Loops from each row are pulled through those of the previous row. The two primary directions in a knit are called the *course* and the *wale*, with the course traveling in the direction of a single row of loops and the wale traveling in the direction of the stack of loops. Figure 2.6-b shows an example knitting pattern. Knitted fabrics are not the main focus of this thesis although the use of micro CT imaging introduced in Chapter 4 can be modified to handle this type of material.

CHAPTER 3

PRIOR WORK

In computer graphics, there has been a large body of work related to modeling and reproducing the appearance of fabrics as well as general translucent media. This chapter summarizes previous work on light transport theory, fabric appearance modeling, physically based rendering of scattering media, and example-based synthesis.

3.1 Radiative transfer

Radiative transfer is used in many areas including astrophysics, neutron transport, and computer graphics [7, 40]. Recently, Jakob et al. [43] generalized this framework to better handle anisotropic volumetric media (including fabrics) and proposed the "microflake model" for phase functions in such media.

In Chapters 4, 5 and 6, we use this more general anisotropic framework; Chapter 7, in contrast, focuses on the classical (isotropic) radiative transfer theory.

Similarity theory. Similarity theory was introduced by Wyman et al. [105, 106] in applied physics. The authors derived a set of relations between two sets of scattering parameters so that the resulting RTEs have identical solution radiance fields when their directional frequencies are bounded. A highly simplified order-1 form of this theory is used extensively in diffusion methods and has been applied to accelerating Monte Carlo simulation of light transport [8, 25]. However, very limited work has been done, in both computer graphics and applied physics, to utilize such relations at higher orders.

Inverse volume rendering. Inverse rendering methods solve for the material properties in a scene given the desired appearance. Multiple methods have been developed to recover subsurface scattering properties [99, 16, 77, 31]. We show in Chapter 7 that similarity theory can be helpful for solving the inverse volume rendering problem.

3.2 Fabric appearance modeling

Fabric reflectance models. Cloth has perennially appeared in graphics as a source of difficult BRDFs. Westin et al. [102] computed cloth BRDFs by raytracing mesostructure models. Ashikhmin et al. [5] rendered velvet and satin using hand-designed microfacet distributions. Adabala et al. [1] proposed a rendering method for woven cloth based on microfacet theory. Drago and Chiba [20] proposed a method to procedurally model different kinds of woven canvases using spline surfaces. Multiple elaborate models [39, 88] have been presented based on the analysis of yarn or fiber directions in a range of woven fabrics. Each of these methods achieved good appearance relative to the then-current state of the art, but they are all specially hand-designed models for individual materials or specific classes. Lu et al. [63] measured and analyzed reflections from velvet, and Ngan et al. [75] measured some fabrics, including satins, but neither proposed models suitable for rendering.

Our approach, presented in Chapters 4 and 5, is based on a completely general system that only has a volume with fibers as its underlying assumption. Thus, we have few fundamental limitations on what textile or textile-like materials can be handled. Further, by importing volumetric detail from the real world, we can achieve good appearance in closeups, and at silhouettes, edges,

and corners, where surface models appear unrealistically smooth and flat.

Image-based techniques. Because standard surface-oriented models are inadequate for complex thick materials, researchers and practitioners have had to fall back on image-based rendering methods like Bidirectional Texture Functions (BTF), which essentially consist of an exhaustive set of photographs of the surface under all possible illumination and viewing directions [13, 26]. Although BTFs produce realistic results for many otherwise difficult materials, this image-based approach requires a significant amount of storage, and is often not of high enough resolution for sharp BRDF features, and generally fails to capture or predict grazing angles, making silhouettes and edges unrealistic.

Volumetric fabric models. Two prominent early volume appearance models are Kajiya and Kay's fur rendering [49], and Perlin and Hoffert's "hypertexture" [79]. Although it has since become more common to render hair and fur using discrete curves, their results demonstrate the value of volumetric models for complex, barely resolved detail. A similar approach is the "Lumislice" representation [107, 9] which focused on modeling and rendering knitwear. Magda and Kriegman [65] describe a method for acquiring volumetric textures which combine a volumetric normal field, local reflectance functions, and occupancy information. All these approaches need significant modeling effort.

Cloth structure. The geometry of cloth structure has been studied for decades [80, 52] by the textile research community. More recently X-ray tomography, using synchrotron facilities [92, 32] or the rapidly improving micro CT scanners [61, 90], has been used to examine the structure of textiles in several

applications. These studies focus on extracting geometric information related to the material's mechanical properties, but have produced some analysis tools [90] that we use in Chapter 4.

Fabric appearance acquisition. Many techniques have been developed for acquiring spatially varying BRDFs [67, 100, 27, 17, 28]. In particular, Wang et al. [100] and Dong et al. [17] both use BRDFs based on tabulated normal distributions to represent a variety of materials including a Jacquard silk satin. These models do an excellent job of capturing the spatially varying anisotropy of the material, but their resolution is limited to that of the photos used for capture.

3.3 Physically based rendering of translucent media

Monte Carlo methods. The solution of the radiative transfer equation (RTE) by Monte Carlo integration was first introduced to computer graphics by Kajiya and von Herzen [50]. Since then, volume path tracing and its variants have frequently been used to render participating media such as clouds or fog [58, 78] as well as fabrics [43, 111, 112, 110].

In addition, various techniques such as volumetric photon mapping [45, 44] and many-lights methods [76, 97] have been developed, which offer faster convergence than path tracing methods, but often at the cost of introducing bias in the results.

Diffusion methods. Diffusion methods replace the RTE with the diffusion equation (DE) by applying a first-order approximation to directional radiance [40]. Jensen et al. [47] introduced a dipole model to graphics which provides an

approximated analytical solution to the DE. This work uses the homogeneous isotropic assumption, in addition to assuming a semi-infinite flat slab geometry. The layered subsurface scattering approach of Donner and Jensen [18] only handles flat layers. A follow-up work [19] removes the flat slab assumption by placing multiple-scattering point sources along a light ray through the medium. D'Eon [15] introduced a more accurate modification of diffusion theory that works better for highly absorbing materials.

An alternative to approximated analytic methods is to solve the DE by finite differences [91, 99] or finite elements [2].

Diffusion methods require the resulting radiance field to be smooth, which is usually violated near material boundaries and in optically thin regions. Consequently, hybrid techniques [60, 19, 34] combine Monte Carlo methods and diffusion for better accuracy.

Diffusion methods do not easily apply to our fabric models as they are highly heterogeneous and anisotropic. Solving the DE over these volumes would require billions of simple finite elements.

Methods approximating higher bounces. Several existing solutions share our approach (Chapter 6) of splitting out the first few path segments, and then looking up a radiance approximation. Volumetric photon mapping [46], also extended to anisotropic scattering from blond hair by Moon and Marschner [70], approximates radiance by using density estimation after a few Monte Carlo bounces. The problem with density estimation for rendering high-resolution microgeometries is that to get enough photons of all necessary orientations and path lengths, the radius of lookup has to be orders of magnitude larger than the underlying microgeometry, thus being locally inaccurate. Photon beams and the

beam radiance estimate [44] can be very useful in optically thin media; however, the mean free path in cloth is a few microns, so the usable length of a beam would likely become much smaller in our case.

The follow-up works by Moon et al. [71, 72], and the related hair rendering approach of Zinke et al. [115], are also based on using approximate radiance after several path-traced bounces. These methods do not take advantage of the modularity from repeated structures, and may not be scalable to the complexity of our volumes. Schroeder et al. [89] render fabrics by replacing actual microgeometry for higher bounces by randomly selected and oriented fibers; however, the required path lengths remain the same, and the evaluation of the model is also quite expensive.

Precomputed approaches. Modular radiance transfer [62] was introduced for diffuse indirect illumination for blocked interiors often found in computer games, and targeted at real-time performance. The key idea is that light needs to be propagated from a surface to a block boundary, then within blocks, and finally back to a surface. We take significant inspiration from this approach, and show how to derive a modular formulation in the domain of high-quality volume rendering in Chapter 6.

The Lumislice approach [108] precomputes scattering within a yarn of cloth, and produces convincing results for fabrics with thick yarns, but does not easily extend to fabrics modeled at the fiber level, or with strong anisotropic highlights. Premože et al. [84] proposed an approach to compute multiple scattering with a "light attenuation volume" precomputed for each light source. This method focuses on thin media like fog and cannot capture highly anisotropic scattering.

Other methods. Rushmeier and Torrance [87] integrated volume scattering in thin participating media into a radiosity framework. This technique formulates light transport using finite elements, related to our approach described in Chapter 6 for a single block, but is not modular and does not scale to high-resolution volumes since block-level precomputation is impossible. Narasimhan et al. [74] derived a general formula to solve the RTE. Their formula, however, applies only to homogeneous participating media and thus cannot be used to solve our problem. Fattal [23] presented an approach based on the Discrete Ordinates Method to solve the RTE approximately. This technique is able to render heterogeneous materials like marble, but is not efficient enough to handle volumes with trillions of voxels.

3.4 Example-based synthesis

Example-based texturing. Example-based texture synthesis techniques create a large output texture using small exemplars. Those algorithms can be performed based on pixels [36, 21, 101, 59] or patches [11, 22, 57, 104] and can also synthesize solid textures [53]. Some texture synthesis algorithms take additional constraints [4, 56, 85], but the forms of such constraints are quite different from those in Chapter 5. Many approaches, including [4], also aim at preserving continuity across synthesized pixels. But the optimizations are normally performed locally and do not ensure global continuities.

Synthesizing appearance/geometry. Several approaches synthesize polyhedral meshes [69], voxelized volumes [114], and appearance models [93, 10] onto arbitrary surfaces. However, all these methods focus on a very different prob-

lem: extending complex exemplars over a non-trivial domain. The technique proposed by Zhou et al. [114], in particular, includes a deformation step to solve a similar problem as the one introduced in Section 5.4.5. Unfortunately, this method needs to be performed at the voxel level and thus does not scale to the size of our problem.

Discrete element synthesis. Recently, techniques that fill a volume with a set of discrete elements have been proposed [38, 64]. Like texture synthesis, these methods usually do not support constraints, or they take constraints that are quite different from ours. And as the name suggests, the basic elements in these methods are "disconnected" from each other, which is not the case in our problem.

CHAPTER 4

BUILDING VOLUMETRIC FABRIC MODELS USING MICRO CT IMAGING

In this chapter, we present the first component of our main pipeline (Figure 1.3): a fundamentally new approach to automatically build volumetric fabric models with micron-resolution.

A fabric's appearance depends heavily on how the component yarns and fibers are arranged, but such arrangement can be highly complicated and thus difficult to describe procedurally. We acquire this structural information using micro CT imaging. Since the CT scans lack optical information, we combine them with a photograph of the sample to obtain a complete volumetric appearance model. The models created by our method lead to fabric renderings with unprecedented fidelity. This work originally appeared at ACM SIGGRAPH 2011 [111] and will be published as a research highlight at Communications of the ACM (CACM).

4.1 Introduction

The appearance of materials like cloth is determined by 3D structure. Volume rendering has been explored for decades as an approach for rendering such materials, for which the usual surface-based models are inappropriate [49, 79, 107]. Recent developments [43] have brought enough generality to volume scattering that we can begin to render fully physically-based *volumetric appearance models* for cloth, fur, and other thick, non-surface-like materials. However, a fundamental problem remains: creating these volumetric models themselves. For surfaces, texture maps derived from photographs are simple and effective, but volumes

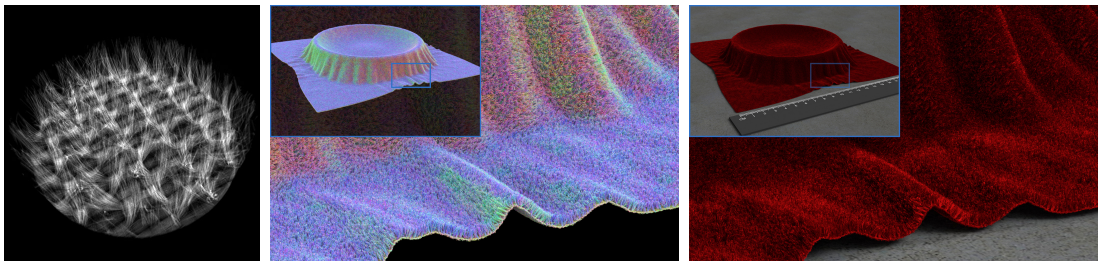


Figure 4.1: We build **volumetric appearance models** of complex materials like velvet using CT imaging: (left) CT data gives scalar density over a small volume; (center) we extract fiber orientation (shown in false color) and tile larger surfaces; and (right) we match appearance parameters to photographs to create a complete appearance model. Both fine detail and the characteristic highlights of velvet are reproduced.

are not so easy. Previous work has primarily relied on procedural methods for modeling volume density, but this has limited generality: significant creative effort is needed to design special algorithms for each new material. Further, these models often miss the subtle irregularities that appear in real materials.

This chapter explores an entirely different approach to building volume appearance models, focusing particularly on cloth. Since cloth's detailed geometric structure is so difficult to model well, we use volume imaging to measure structure directly, then fill in optical properties using a reference photograph. We do this by solving an inverse problem that statistically matches the texture between photographs and physically based renderings (which include global illumination and multiple scattering). We focus on textiles because they exhibit a wide range of appearance, but share a common basic structure of long, shiny fibers. Textile rendering is important for many applications, but is challenging because cloth is structured, causing complicated textures and reflectance functions, yet irregular, causing difficult-to-model randomness. The thick, fuzzy nature of cloth makes volume models a good fit, if only there were a general solution for constructing them.

Many volume imaging technologies have been developed, including computed tomography (CT), magnetic resonance, ultrasound, and others. But unlike photographs, the resulting data does not directly relate to the optical appearance of the material; only to its structure. As a result, volume renderings of these images are useful for illustrating hidden internal geometry, but not directly for rendering realistic images. For instance, a micro CT scan of woven cotton cloth gives a detailed view of the interlaced yarns and their component fibers, showing exactly how the fibers are oriented and how the yarns are positioned, but no information about how they interact with light: there is no way to tell whether the fabric is black or white or any color in between.

We show in this chapter that remarkably little additional information is required to extend CT data to a realistic appearance model. The value of knowing 3D structure is obvious for rendering close-up views where these details are visible. But equally importantly, the shape and arrangement of fibers in the material also determines the overall appearance of the material — the shape and quality of specular highlights, and how the visual texture varies with illumination and view. When coupled with the right rendering technology, a simple local model of reflection from fibers automatically predicts the characteristic appearance of very different materials like velvet and satin, simply by knowing the 3D structure of the material.

The contribution of this work is to show how to enhance the structural information from a CT scan of a small sample of fabric by combining it with appearance information from a photograph of the material to construct plausible and consistent optical properties that produce realistic appearance when rendered using a physically based volume renderer. We describe our end-to-end volume appearance modeling pipeline and demonstrate it by acquiring models of cloth

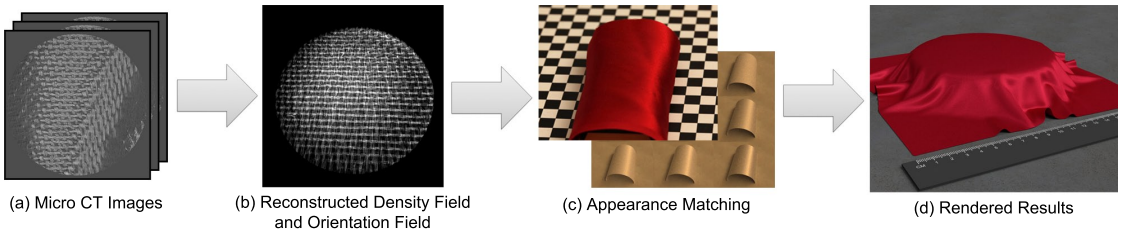


Figure 4.2: **Our volume appearance modeling pipeline.** (a) CT images are acquired, (b) the density field and orientation field of the volume are created, and (c) optical parameters of the volumetric model are assigned by matching statistics of photographs with rendered images. (d) Larger models are rendered using our acquired volumetric appearance and geometry models.

with very different appearance, ranging from matte to shiny and textured to smooth, capturing their characteristic highlights, textures, and fuzziness.

4.2 Overview

The goal of our system is to create realistic volumetric appearance models of cloth. We need to generate a sampled 3D volume that describes the optical properties of the material at each voxel so that, when rendered with a physically based rendering system, it realistically reproduces the appearance of real cloth.

Because cloth is made of fibers, we need a volume scattering model that can handle the anisotropy of fibers; we chose a modified version of the model proposed by Jakob et al. [43] (detailed in Section 4.3) for this purpose. This model requires an optical density, an albedo, and two phase function parameters: an orientation vector and a specular lobe width.

Our technique begins with a micro CT scan of a small area of material, showing detail at the level of individual fibers over a fraction of a square centimeter. Such scans can readily be ordered at moderate cost (a few hundred US dollars) from a number of facilities, and suitable desktop CT scanners are becoming avail-

able. In a sequence of three stages (Figure 4.2) we process and augment this data, ending with a volume that defines the required scattering model parameters using density and orientation fields derived from the CT data, plus three global parameters: the albedo, the lobe width, and a density multiplier that scales the density field.

The first stage (Section 4.4) processes the density volume to augment it with orientation information and to remove noise by convolving the data with 3D oriented filters to detect oriented structures, and thresholding to separate meaningful structure from noise. This stage produces the density and orientation fields.

This volume can be rendered only after the global optical parameters are determined. The second stage (Section 4.5) makes use of a single photograph of the material under known (but not controlled) lighting, and associates optical properties with the oriented volume from the first stage by matching the texture of the rendered volume to the texture of the photograph.

The resulting volume model is good for rendering small samples; the third stage takes this small patch and maps it over a large surface of cloth, using randomized tiling to replicate the material and shell mapping [83] to warp it.

The resulting renderings (Sections 4.6 and 4.7) show that this unique approach to appearance modeling, leveraging direct information about mesoscale geometry, produces excellent appearance from the small scale, where the geometry itself is visible, to the large scale, where the directional scattering properties naturally emerge from the measured 3D structure. The characteristic appearance of difficult materials like velvet and satin is predicted by our rather minimal volume scattering model, even though we use no light scattering measurements

that could tell these materials apart, because accurate geometric information is available.

4.3 Fiber scattering model

We model light transport using the anisotropic radiative transfer equation (2.9) from Jakob et al. [43]. This equation is a generalization of the isotropic RTE (2.1) that adds support for a directionally varying amount of "interaction" with a medium. For instance, the directional dependence of $\sigma_t(\boldsymbol{\omega})$ is necessary to model the effect that light traveling parallel to coherently aligned fibers faces less obstruction than light traveling perpendicular to the fibers.

To specify the problem to be solved, we must choose a compatible scattering model that will supply internally consistent definitions of σ_t, σ_s, and f_p. For this purpose, we use the *micro-flake* model proposed in the same work. This volume analogue of microfacet models represents different kinds of volume scattering interactions using a directional *flake distribution* $D(\mathbf{m})$ that describes the orientation $\mathbf{m}$ of (unresolved) idealized mirror flakes at every point in space. Similar to microfacet models, the phase function then involves evaluating $D(\mathbf{m})$ at the half-way direction between the incident and outgoing direction. For completeness, we reproduce the model's definition below:

$$\sigma_t(\boldsymbol{\omega}) = a\,\rho \int_{S^2} |\boldsymbol{\omega}\cdot\mathbf{m}| D(\mathbf{m})\,\mathrm{d}\mathbf{m}$$

$$\sigma_s(\boldsymbol{\omega}) = \alpha\,\sigma_t(\boldsymbol{\omega})$$

$$f_p(\boldsymbol{\omega}'\to\boldsymbol{\omega}) = \frac{a\,\rho\,\alpha}{4\sigma_s(\boldsymbol{\omega})}\left[(D(\mathbf{h}(\boldsymbol{\omega},-\boldsymbol{\omega}')) + D(-\mathbf{h}(\boldsymbol{\omega},-\boldsymbol{\omega}'))\right]$$

Here, ρ denotes the particle density, a is the area of a single flake, α is the associated albedo, and $\mathbf{h}(\boldsymbol{\omega},\boldsymbol{\omega}') \coloneqq (\boldsymbol{\omega}+\boldsymbol{\omega}')/\|\boldsymbol{\omega}+\boldsymbol{\omega}'\|$. Note that the above expressions

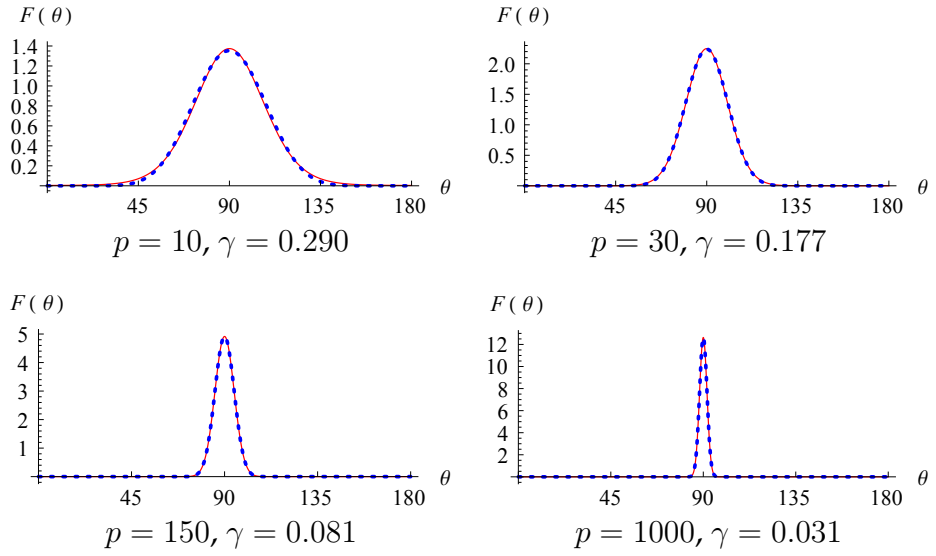


Figure 4.3: **Comparison** between the $\sin^p \theta$-type distribution with exponent p (blue) and our Gaussian-type distribution (red) with standard deviation γ.

are simplified by assuming the flakes have albedo independent of the scattering angle. This reduces our search space considerably and still leads to a model that can represent scattering interactions with a variety of fibrous materials reasonably well.

To simulate scattering from a rough fiber with direction $\boldsymbol{\omega}_f$, Jakob et al. propose the flake distribution $D(\boldsymbol{\omega}) = c_0 \sin^p(\boldsymbol{\omega}_f, \boldsymbol{\omega})$, where higher values of p correspond to smoother fibers and c_0 is a normalization constant. This model leads to flake normals concentrated near the plane perpendicular to $\boldsymbol{\omega}_f$; the underlying motivation is to represent the normal directions observed on the original fiber's surface, which predominantly point in these directions.

4.3.1 Alternative flake distribution

One serious drawback of the $\sin^p$-type distribution is that most integrals over it do not have a closed form. This is problematic, since it effectively prevents

the use of the inversion method for generating random samples distributed according to D. Since our rendering pipeline crucially depends on this ability (see Section 4.6), we propose an alternative flake distribution that is convenient to integrate, while capturing the same key feature of the $\sin^p$ model, namely that it is primarily concentrated perpendicular to the fiber direction.

We use the following density function, which specifies a truncated Gaussian centered around the great circle perpendicular to $\boldsymbol{\omega}_f$:

$$D(\boldsymbol{\omega}) = \frac{1}{(2\pi)^{3/2}\,\gamma\,\mathrm{erf}\left(\frac{1}{\sqrt{2}\gamma}\right)} \exp\left(-\frac{(\boldsymbol{\omega}_f \cdot \boldsymbol{\omega})^2}{2\gamma^2}\right)$$

where the standard deviation γ determines the roughness of the fiber and $\boldsymbol{\omega}_f$ denotes the fiber direction. The model captures the same qualitative behavior as the $\sin^p$ model over a large range of parameter values (Figure 4.3).

To summarize, the parameters required to create renderings are:

- $\boldsymbol{\omega}_f$, the local fiber orientation,
- γ, the standard deviation of the flake distribution,
- α, the single scattering albedo of the flakes,
- a and ρ, the area and density of micro-flakes. Their product roughly corresponds to the interaction coefficient σ_t in traditional isotropic volume rendering, and we therefore set them to a multiple of the processed CT densities, i.e. $a\rho(\mathbf{x}) \coloneqq d \cdot \mathrm{CT}(\mathbf{x})$, where d is a constant of proportionality.

Section 4.4 discusses the steps needed to obtain $\mathrm{CT}(\mathbf{x})$ and $\boldsymbol{\omega}_f(\mathbf{x})$. In Section 4.5, we describe how to find α, γ, and d, and Section 4.6 explains how to use our scattering model in Monte Carlo rendering.

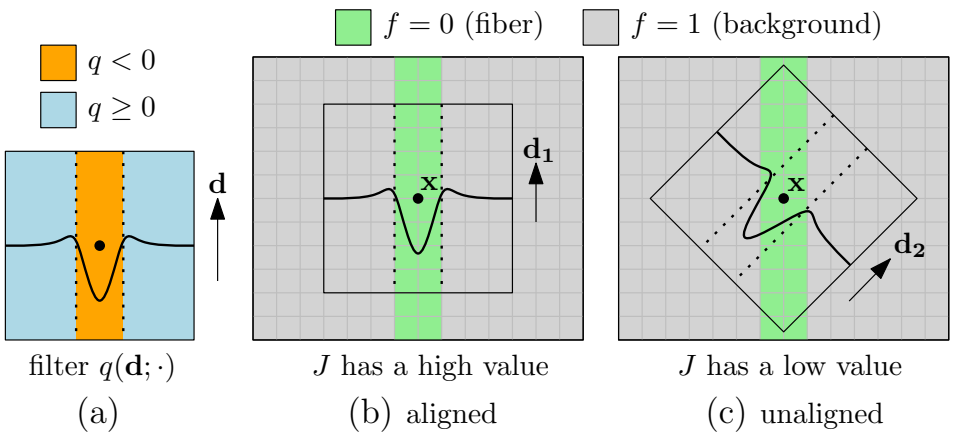


Figure 4.4: **Computing function J in 2D:** (a) shape of the filter q; (b) when q is aligned to the fiber; (c) when q is unaligned.

4.4 CT image processing

Micro CT (computed tomography) devices, which use X-ray CT methods to examine small to microscopic structures, are increasing in availability, and this imaging modality is suited to a wide range of materials from which a small sample can be extracted for scanning.

In this section we describe the process of extracting fiber orientation from the CT density volume using a special fiber-detecting filter. Following this, we explain the processing steps needed to obtain orientation and density fields suitable for rendering.

4.4.1 Recovering the orientation field

CT images provide a voxelized density field with no direction information. Since our optical model requires an orientation for the phase function, it is necessary to reconstruct an orientation for every non-empty voxel. Our approach uses oriented filters to detect fibers, based on similar filters used by Shinohara et al. [90]

to locate fibers in CT data. We chose this approach because of its demonstrated application to fiber detection in CT data, though alternatives [6] are possible.

To detect a fiber with orientation $\mathbf{d}$ at location $\mathbf{p}$, Shinohara proposes a cylindrically symmetric filter oriented with the axis $\mathbf{d}$, consisting of a difference of Gaussians in distance from the axis:

$$q(\mathbf{d}; \mathbf{p}) \coloneqq -2\exp(-sr^2) + \exp(-tr^2)$$

where $r = \|\mathbf{p} - (\mathbf{p} \cdot \mathbf{d})\mathbf{d}\|$ is the distance from the filter's axis and the parameters s and t (normally $s < t$) are empirically adjusted based on the size of the fibers present in the sample (see Figure 4.4).

The raw CT volume is thresholded at a value ϵ_d, resulting in a binary volume f where

$$f(\mathbf{x}) := \begin{cases} 0 & \mathrm{CT}_{\mathrm{raw}}(\mathbf{x}) \geq \epsilon_d, \\ 1 & \mathrm{CT}_{\mathrm{raw}}(\mathbf{x}) < \epsilon_d. \end{cases}$$

Then f is convolved with the filter q for each of a fixed set of orientations:

$$J(\mathbf{x}, \mathbf{d}) := \sum_{\mathbf{p} \in V} q(\mathbf{d}; \mathbf{p}) f(\mathbf{x} + \mathbf{p}) \tag{4.1}$$

where V is a cubic volume of edge length h. For parameter values, refer to Table 4.1.

As shown in Figure 4.4, the function J reaches a maximum value when $\mathbf{d}$ equals the fiber's orientation. So the orientation field is computed by finding, for each voxel $\mathbf{x}$, the $\mathbf{d}'$ that maximizes $J(\mathbf{x}, \mathbf{d}')$ and setting $\boldsymbol{\omega}_f(\mathbf{x}) = \mathbf{d}'$. In our implementation, we pre-compute q on a set of directions $\{\mathbf{d}_i\}$ picked from a $32 \times 32 \times 6$ cubemap. Then for each non-empty voxel $\mathbf{x}$, we set $\boldsymbol{\omega}_f(\mathbf{x}) = \mathbf{d}_j$ where $j = \arg\max_i J(\mathbf{x}, \mathbf{d}_i)$.

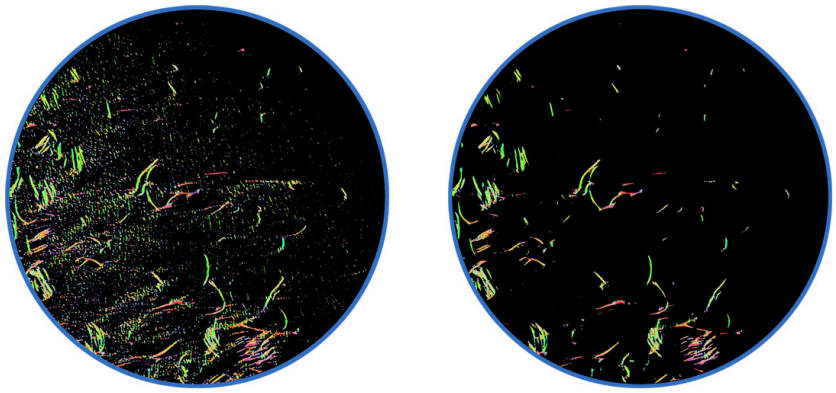


Figure 4.5: **Computed orientation field** for a piece of gabardine with each direction (x, y, z) mapped to RGB color $(|x|, |y|, |z|)$. Left: without thresholding on J; right: with thresholding on J.

4.4.2 Denoising CT images

The CT images usually contain considerable amounts of noise, particularly for low-density materials like our cloth samples, and removing the noise is critical for obtaining good quality data for rendering. Since cloth structure is always oriented, and the noise is generally fairly isotropic, the value of J is useful in noise removal.

In our system we use two thresholds to remove noise. The first threshold ϵ_d is on the voxel values themselves, and is used to remove faint background noise that would otherwise cloud the model. This thresholding creates the binary volume f. The second threshold ϵ_J is on the value of J and is used to remove isotropic noise that has density values that are too high to remove by the first threshold. We set

$$\mathrm{CT}(\mathbf{x}) := \begin{cases} \mathrm{CT}_{\mathrm{raw}}(\mathbf{x}) & \mathrm{CT}_{\mathrm{raw}}(\mathbf{x}) \geq \epsilon_d \text{ and } J(\mathbf{x}, \boldsymbol{\omega}_f(\mathbf{x})) \geq \epsilon_J; \\ 0 & \text{otherwise.} \end{cases}$$

Figure 4.5 shows the significance of adding this second threshold.

4.4.3 Data replication

The volume data needs to be replicated for rendering since our samples are very small (no larger than $0.5 \times 0.5\ \mathrm{cm}^2$). We will explore example-based synthesis in Chapter 5 which provide sophisticated tools to do this. Here we consider two simple randomized tiling methods to cover the surfaces with tiles of volume data drawn from our models without introducing distracting regular structures. In both methods the surface is simply covered by a rectangular array of tiles copied from the volume, without continuity at the tile boundaries.

For materials without visible regularity, such as velvet and felt, each tile on the surface is copied from a rectangular region centered in the volume. To provide variation in local structure, for each tile this source rectangle is rotated by a different random angle. For materials with woven structure, like silk and gabardine, we use a similar approach, but use random translations of the source tile instead of rotations. The weave pattern in each sample is manually identified and a rectangular area is marked that contains an integer number of repeats. Then each (smaller) surface tile is chosen from a subrectangle that contains a matching section of the weave. The result is a tiling that reproduces the correct weave pattern and avoids obvious repeating of texture. We then map the tiled data to arbitrary surfaces using shell mapping [83].

4.5 Appearance matching

Processing the CT data yields the spatially varying density and orientation for the volume. But the optical appearance parameters of the model remain to be determined. Since the CT scan does not give us the material's optical proper-

ties, we make use of a photograph of the material to compute the appearance parameters.

To make the problem tractable, we assume that the volume contains the same material, with differences only in density and orientation. This is appropriate for fabrics made from a single type of fiber, which encompasses many important examples. Fabrics containing yarns of different materials are future work. Thus, the appearance parameters that must be determined are the same across the whole volume. They are: the standard deviation of the flake distribution γ (corresponding to fiber roughness), the scattering albedo α (corresponding to material color), and the density scale d (corresponding to opacity). Figure 4.6-(a) illustrates the effects of these parameters.

To match the material's optical properties, we must use photographs of the sample. One approach is to photograph the same sample that was scanned, calibrating the camera to the scan and associating pixels in the image with rays in the volume. This calibration and acquisition is non-trivial; the fine resolution of the scans poses practical difficulties. Further, we found that this level of detail is not required to determine the small number of parameter values we seek. Instead, we assume that the fabric is statistically similar across different patches. Thus, our approach is to statistically match the texture of rendered images with a photograph of a different section of the same cloth under uncontrolled but known lighting.

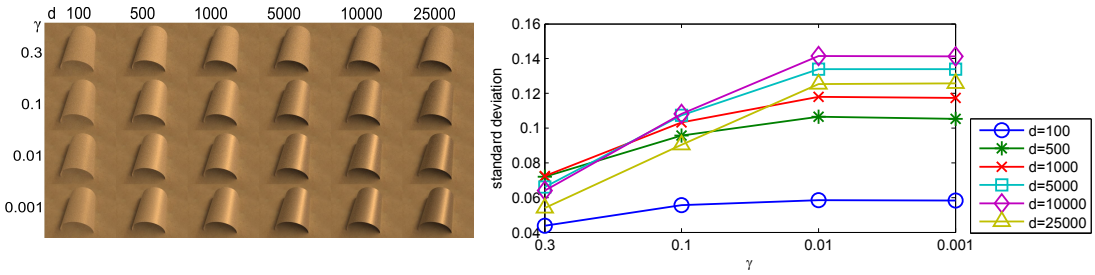
We now describe the metrics we use to match the optical parameters to the photograph, and then describe our matching algorithm.

4.5.1 Metrics for matching

Appearance matching is not a straightforward process of mapping colors from the photos into the volume, because the volume model describes local scattering properties, but the appearance is defined by a global volumetric multiple scattering process. Our approach is to repeatedly render the volume using our physically based renderer, and adjust the optical parameters to match certain texture statistics of the rendered images to statistics of the photograph.

We match two simple statistical measures: the mean pixel value and the standard deviation of pixel values, computed over corresponding regions of a photograph and a rendering of approximately similar geometry. This approach effectively matches the image brightness and texture contrast in the matching region. We tried measures such as the CDF of intensities [36] and skewness [73], but found that the mean and standard deviation measures were simpler and robust. Thus, the only information that flows from the photograph to the volume model is the mean and standard deviation of pixels in a single rectangle.

The appearance matching process involves choosing the geometry, camera position, lighting, and matching region. These are inherently manual choices, and we used the principle of choosing a setup that shows the distinctive features of the cloth's appearance. For instance, we made sure to use a configuration where the highlight was visible on the satin. Beyond this we did not take any special care in arranging the appearance matching inputs, and the results do not appear to be sensitive to the details.



(a) (b)

Figure 4.6: (a) Renderings of a cylinder tiled with the satin volume, with fixed albedo and varying lobe width γ and density mlutiplier d. (b) The corresponding standard deviation of pixel values for the satin sample: sharper lobes provide shinier appearance and result in greater standard deviation. The role of d is more complicated.

4.5.2 Optimization procedure

As shown in Figure 4.6, the density multiplier plays a fairly complicated role with respect to both measures. Given that our forward process, which is essentially Monte Carlo path tracing, is quite expensive, we chose to pre-determine the density multiplier in our implementation by rendering such a matrix. Fixing the density multiplier simplifies the inverse problem and leads to a practical solution. We found that the algorithm is not particularly sensitive to the choice of density multiplier; our results use two main settings which differ by an order of magnitude (see Table 4.1).

With a fixed density multiplier, we solve for the values of albedo (α, estimated separately in red, green, and blue) and lobe width (γ, a single scalar value) using an iterative algorithm. Note that the mean and standard deviation of pixel values change monotonically with changes in α and γ respectively[1]. Thus, a binary search can be used to significantly improve performance as follows: first, an initial guess of γ is assumed, and we search for the α to match

[1]This holds as long as γ exceeds a minimum value ($\gamma = 0.01$ for all our experiments); below this value the variance of fiber orientations limits glossiness.

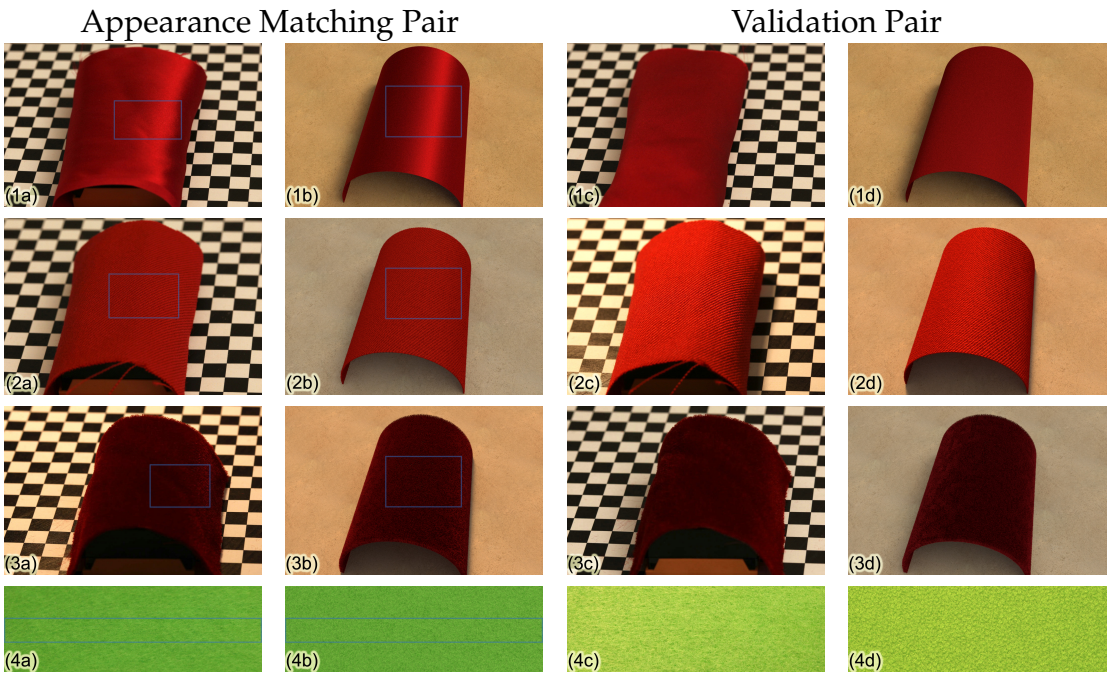


Figure 4.7: **Appearance matching results** for (from top to bottom) (1) silk, (2) gabardine, (3) velvet, and (4) felt. Columns (a) and (c) show photographs of the materials, and (b) and (d) show rendered images. The left two columns form the appearance matching pair, in which the blue boxes indicate manually selected regions for performing our matching algorithm. The right two columns, the validation pair, validate our matches qualitatively under different configurations.

the mean pixel value. Then, fixing α, we perform a search for the γ to match the standard deviation. These iterations are repeated until a match is found. In practice, this approach converges quickly, usually in 2 or 3 iterations.

Finally, we take another photo under a different setup and render a corresponding image as a qualitative validation (see Section 4.7). Figure 4.7 shows the appearance matching results for four different materials.

Material	Data Size	s	t	h	ϵ_d	ϵ_J	d	γ	α
Gabardine	$992 \times 1012 \times 181$	1	2	16	0.45	–10	5000	0.1	$(0.892, 0.063, 0.048)$
Silk	$992 \times 1013 \times 46$	3	4	12	0.4	–6	5000	0.01	$(0.699, 0.030, 0.080)$
Velvet	$992 \times 1012 \times 311$	3	4	12	0.4	–1	500	0.1	$(0.555, 0.040, 0.074)$
Felt	$992 \times 1012 \times 485$	1	2	16	0.4	–30	500	0.125	$(0.518, 0.915, 0.365)$

Table 4.1: **Fiber filter and scattering model parameter values for our material samples:** s and t are the shape parameters, and h is the volume size of the filter (Section 4.4); ϵ_d and ϵ_J are the noise thresholds. The optical parameters include d, the density multiplier, and the parameters found by our appearance matching algorithm: γ, the standard deviation of the flake distribution, and α, the single-scattering albedo.

4.6 Rendering

We render all our scenes using a basic Monte Carlo path tracer, which handles the directionally varying properties of the medium described in Section 4.3. One important part of this process entails generating samples from the phase function $f_p(\omega' \rightarrow \omega)$. In this section, we adopt the notation of the integral form of the RTE, i.e. ω is held fixed, and we are interested in sampling the direction ω', from which to gather illumination.

In prior work, [43] used a spherical harmonics representation for this purpose. However, this approach has several undesirable properties. First, a sample weight is needed to account for the fact that the sampling routine is only approximate, which increases variance. Second, computing the spherical harmonics coefficients is a time-consuming process, which needs to be repeated for any change in the model parameters. This is problematic, since our fitting stage explores many different parameter sets. Most importantly, the spherical harmonics approach suffers from ringing and therefore cannot handle some of the highly specular material configurations in our parameter space.

In the following section, we first propose a naïve sampling method, which is

not directly usable due to its high variance. We then demonstrate how rejection sampling can be used to turn the naïve method into an exact sampling scheme.

4.6.1 Alternative sampling strategy

In the surface case, importance sampling for microfacet models often takes the approach of sampling a microfacet normal, then using it to compute an outgoing direction [96]. Additional factors, such as the Fresnel reflectance and the Jacobian of the direction mapping must be accounted for in a weight associated with the sample. If we apply this approach to the micro-flake model, we obtain a sampling strategy with the following density:

$$f_1(\omega' \to \omega) = \frac{D(h(\omega, -\omega'))}{2\,|\omega' \cdot h(\omega, -\omega')|}$$

where the denominator is the aforementioned Jacobian (an extra factor of 2 is required in comparison to the surface case, since micro-flakes reflect from both sides). The sample must be assigned the weight:

$$w_1(\omega' \to \omega) = \frac{f_p(\omega' \to \omega)}{f_1(\omega' \to \omega)} = \frac{a\rho}{\sigma_t(\omega)}\,|\omega' \cdot h(\omega, -\omega')| \tag{4.2}$$

where we have assumed without loss of generality that $D(\omega) = D(-\omega)$. Equation 4.2 indicates a fundamental problem of this approach, namely that w_1 can become large when $\sigma_t(\omega) \approx 0$. This becomes a fundamental limitation as specularity increases, particularly when there are many long transport paths (weights are multiplicative along paths and can build up).

To deal with this problem, we use rejection sampling to derive a sampling strategy that generates samples exactly according to f_p. Since $a\rho/\sigma_t(\omega)$ is an upper bound on f_p/f_1, the following algorithm produces samples of the desired density:

```
function SAMPLE-f_p(ω)
    loop
        m ← SAMPLE-FLAKE-DISTRIBUTION()
        Draw ξ ~ U(0,1)
        if ξ < |ω · m| then
            return 2(ω · m)m − ω
        end if
    end loop
end function
```

The added cost of rejection sampling is very small — in our example scenes, only 2-3 iterations were required on average. The approach described in this section is general and might also be useful to improve sampling techniques that are traditionally used for microfacet reflectance models.

4.6.2 Sampling the flake distribution

The algorithm above assumes the availability of a routine SAMPLE-FLAKE-DISTRIBUTION that can draw samples distributed according to $D(\omega)$. We apply the inversion method in spherical coordinates to the distribution described in Section 4.3, and find that the latitude (θ) component integrates to:

$$F(\theta) := \frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{\cos\theta}{\sqrt{2}\gamma}\right) \Big/ \operatorname{erf}\left(\frac{1}{\sqrt{2}\gamma}\right)\right)$$

where, $F(0) = 0$, $F(\pi) = 1$, and we have temporarily assumed that $\omega_f = (0, 0, 1)^T$. To sample θ, we find $F^{-1}(\xi_1)$ numerically using Brent's method, where ξ_1 is uniformly distributed on $[0, 1]$. About 10-18 iterations are required to arrive at machine precision. For the longitude (φ) component, we set $\varphi = 2\pi\xi_2$ (where ξ_2

is another uniform variate). To handle general fiber directions, we rely on the same sampling code and simply apply the appropriate rotations to the incident and outgoing directions.

4.7 Results

Our results are based on samples of silk satin, velvet, felt, and wool gabardine, which were sent to the High-Resolution X-ray Computed Tomography Facility at The University of Texas at Austin. All fabrics were scanned in an XRadia MicroXCT scanner using 1024^3 volumes. The silk stain used a voxel size of 2.5 μm, while the others used 5 μm.

As necessary, our initial data cleanup included corrections to equalize density and contrast between the center and edge of the volume (vignetting). Further, we straightened the slightly non-planar cloth samples using geometric warping, by fitting a second-order polynomial $p(x, y)$ to points distributed proportional to the CT densities and then resampling the whole volume using the mapping $f(x, y, z) = (x, y, z - p(x, y))$. We then ran the CT image processing pipeline (see Section 4.4), with the parameters reported in Table 4.1. Depending on the thickness of the sample, processing took between 1 and 8 hours on a QSSC-S4R Intel Server with 4 Xeon X7560 8-core processors and 32 GB of memory.

Our rendering implementation is based on the open source rendering system Mitsuba [42], which was extended to handle the new micro-flake distribution (Section 4.3). The rendering itself was done on the Amazon Elastic Compute Cloud (EC2), where we used between 8 and 32 `c1.xlarge` instances (each having 8 cores and 7GB of memory) to jointly render the individual images at a resolution of 2.6 megapixels. With 32 instances, rendering times range between

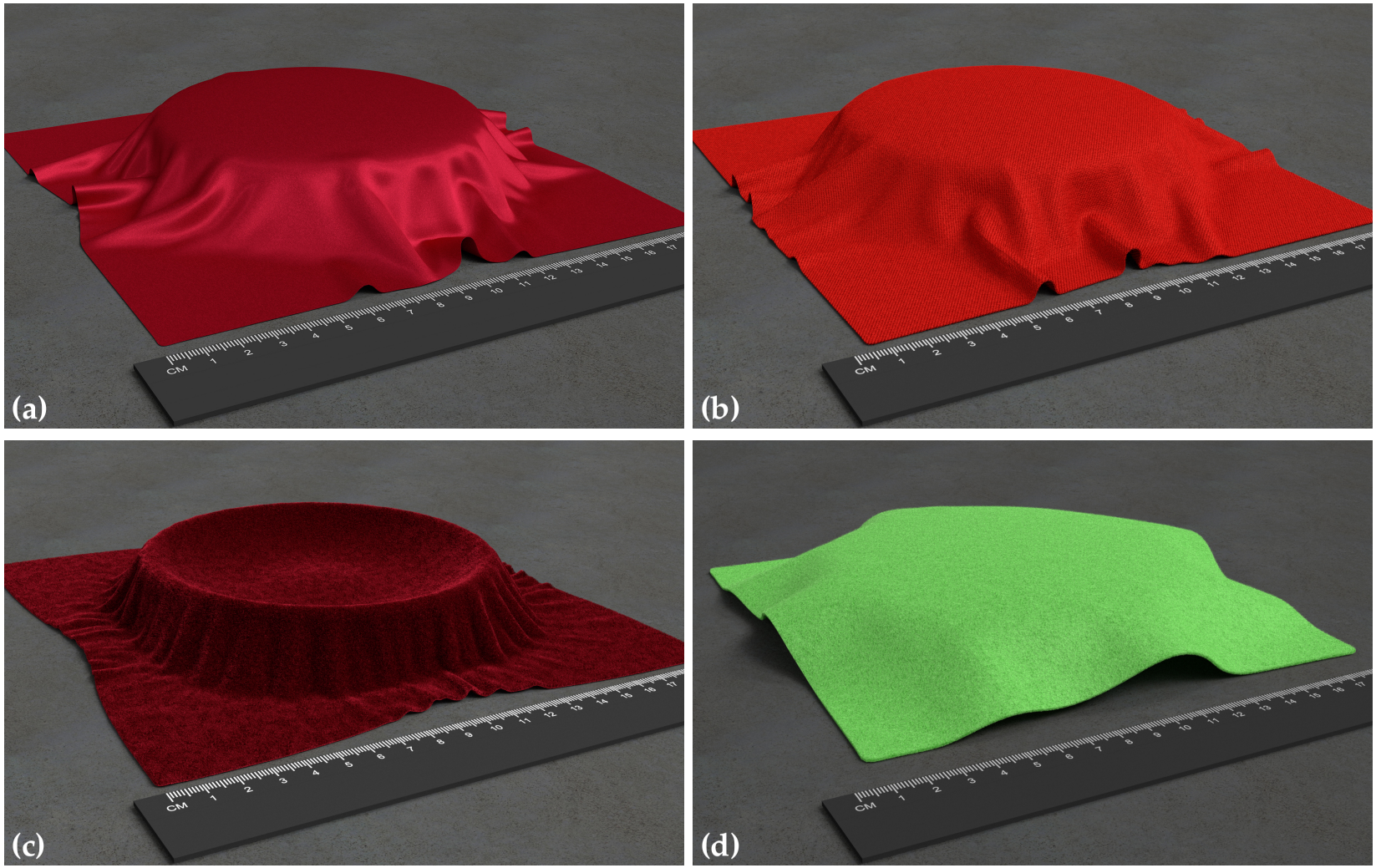


Figure 4.8: **Fabrics in draped configurations with our volumetric appearance model:** (a) silk satin, (b) gabardine, (c) velvet, (d) felt.

3.7 (velvet) and 7.4 (satin) hours per image.

Figure 4.7 shows results obtained by our appearance matching scheme; the left two columns, the *appearance matching pair*, show the image pairs used for appearance matching (the blue rectangle is used for matching), and the right two columns, the *validation pair*, show a different image pair, also with known and matched lighting, to test how well the model generalizes to other configurations. The sizes of all samples we used for appearance matching are roughly 10×10 cm.

Figure 4.8 shows the resulting models shell-mapped onto draped fabric geometry and rendered under environment lighting.

The silk satin (charmeuse) has a structure of mainly parallel fibers on the surface, resulting in a strong anisotropic highlight. In Figure 4.7-(1), the appearance matching pair uses a cylindrically curved piece of material, and the matching region was chosen to include a highlight to allow the matching process to tune γ appropriately. Good results are obtained despite the mismatch between the ideal cylinder in the rendering and the flatter shape of the real material, illustrating that a casual setup suffices. Using the parameters obtained from this view, the validation pair shows the fabric rotated 90 degrees and draped over the same cylinder. At this angle the fabric exhibits almost no highlight; this anisotropic appearance is correctly predicted by our model.

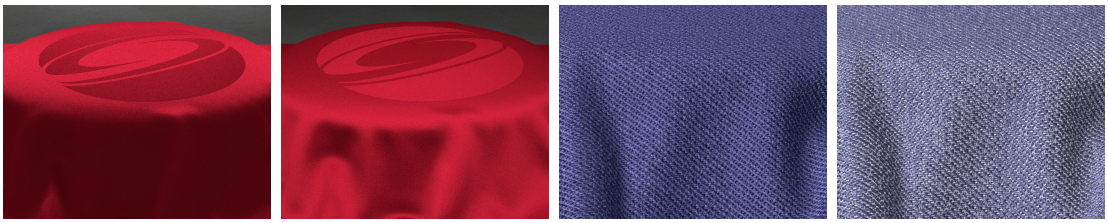
The satin is shown in a draped configuration in Figure 4.8-(a). No reflectance model, BRDF, BTF, or other multi-view image data is used for these renderings — the orientation information in the volume automatically causes the characteristic appearance of this fabric to emerge when the model is rendered.

For gabardine, a wool twill fabric, the variation in texture with illumination

direction is an important appearance characteristic. In Figure 4.7-(2), the appearance matching pair is lit with a low-frequency environment map. The validation pair accurately predicts the texture under a different lighting condition, which involves a strong luminaire at the top. In the draped configuration in Figure 4.8b, the volume model captures subtle foreshortening effects and the silhouette appearance, as well as the subtle variations in texture across the surface. The appearance at the cut edge gives the proper impression of the thickness of the fabric (compare to the very thin satin material), which is a perennial difficulty with surface models.

Velvet, a material with a cut pile (like a carpet), has a visible surface composed of fibers that stick up from the base material. It has a very distinctive appearance, with a characteristic grazing-angle highlight. Appearance matching for velvet (see Figure 4.7-(3)) was done using a curved configuration and the same harsh lighting as used for gabardine's validation, producing distinct highlights on both sides of the cylinder. The validation pair shows a different, softer lighting, which results in a less distinct highlight; our model agrees qualitatively with the photograph. The appearance of velvet depends on how the fibers are brushed, and our random tile rotation method produces randomly brushed velvet. In Figure 4.8-(c), we demonstrate how our model reproduces the characteristic velvet highlights. Further, the edges and silhouettes convey the considerable thickness and weight of this material.

Felt is a non-woven textile consisting of a disorganized layer of matted fibers. The thickness and fuzziness of this material are important appearance attributes that are generally difficult to model and render. Since felt does not exhibit an overall specular highlight, we used a flat patch for appearance matching; because of limited depth of field we limited the matching region to a thin rectangle



(a1) (a2) (b1) (b2)

Figure 4.9: **Renderings obtained by editing the volumetric representation.** (a) The material is flipped using a binary texture map (two lighting conditions are shown). (b) The gabardine sample is rendered with a blue hue (b1); we then detect weft fibers based on their orientation and color them white, which produces a material resembling denim (b2).

where the photograph is in good focus. The illumination conditions for the appearance matching and the validation are the same as those for the gabardine. The color and the contrast due to self-shadowing attributes are matched nicely and generalize well to the second illumination condition. One limitation for this material is that it has substantial low-frequency content in its texture, which our small sample area did not capture in the CT imaging, leading to a slightly more uniform appearance in our tiled material. Figure 4.8-(d) demonstrates the ability of our volumetric appearance model to capture the material's thick, fuzzy appearance.

A 3D, physically based model also allows more meaningful editing than image-based methods. Figure 4.9 shows renderings created after performing simple edits to the underlying volume representation. In the top row, we reflect lookups into the satin volume data across the central plane of the fabric, conditioned on a binary texture map that covers the surface. This edit reveals the back face of the satin weave, which exhibits softer reflections due to less coherent fiber directions, much as these weaves are interchanged in jacquard-woven satin. Two different lighting conditions are shown.

In the bottom row of Figure 4.9, we extend the gabardine model with a spa-

tially varying albedo value. The albedo is computed as a function of orientation, so that fibers in the warp and weft are assigned different colors. With blue warp and white weft a fabric similar to denim is produced, though made of wool rather than cotton.

Finally, we compare our method to the surface-based BRDF and texture model introduced by Irawan [39] (Figure 4.10). For these two examples, Irawan fit his model to BRDF measurements of exactly the same materials we measured. The renderings for Irawan's and our methods took roughly 8 and 64 core hours, respectively.

At the large scale, the BRDFs of the fabrics match reasonably well. Irawan showed that his model matches the measured BRDFs of these materials to a similar degree of fidelity, so this confirms that our method predicts large-scale reflectance from the structure and a single image. For yarn-scale texture, the two models produce generally similar results, though Irawan's model is lower in contrast for the gabardine because it does not account for shadowing. It also produces a more uniform appearance. At the small scale, as seen in the insets, and at silhouettes and edges, our detailed volumetric model produces dramatically more realistic results.

4.8 Conclusion

We have demonstrated a new, multimodal approach to making realistic volume models of cloth that capture both the 3D structure evident in close-up renderings and the BRDF evident in farther-away views. Unlike previous methods for capturing cloth appearance using BTFs, our method explicitly models the 3D structure of the material and, interestingly, is able to capture the directional

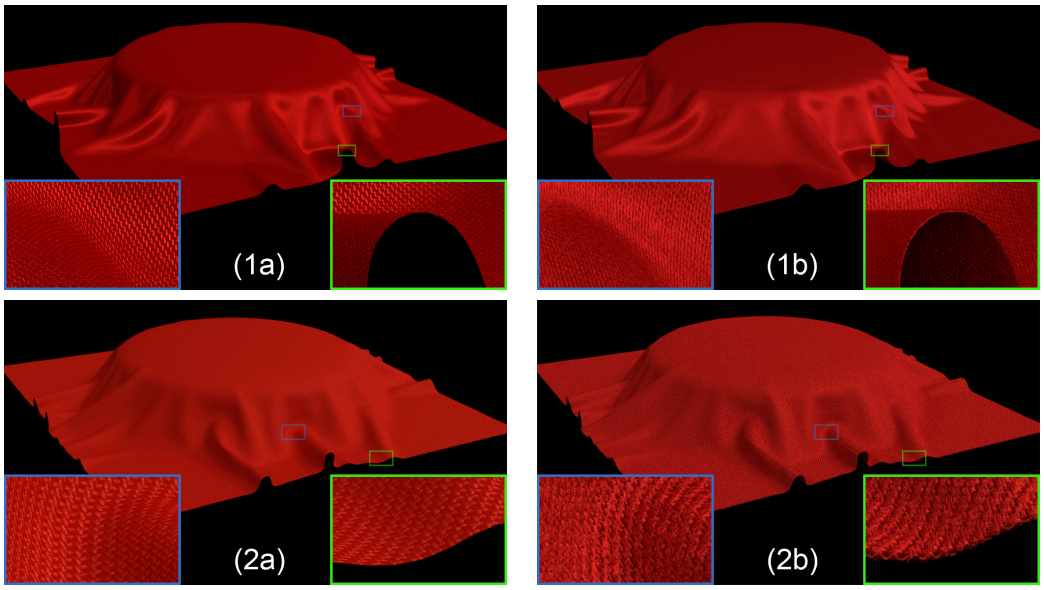


Figure 4.10: Silk satin (1) and gabardine (2) rendered with Irawan's surface-based representation (a) and with our model (b).

reflectance of the material automatically because of this structure.

Our modeling approach uses CT imaging where it is strongest, in measuring 3D structure, and it uses photographs where they are strongest, in measuring color and texture. By matching texture statistics we merge these two sources of information, resulting in a volume model that can produce both close-up views with rich detail of fuzz and fiber structure *and* the characteristic BRDFs (highlights) of these materials that emerge naturally from rendering the measured structure. No BRDF measurements are made, and only a few parameters are adjusted in the optical model. The appearance of the cloth is created by a simple anisotropic phase function model together with the occlusion and orientation information extracted from the volume. This chapter shows that since geometric structure is what creates the complex appearance of textiles, once we acquire the structure, we are most of the way to modeling the appearance.

Aside from its implications regarding how material appearance can be mod-

eled from structure, this is also quite a practical method for appearance modeling. All that is required to model a material is a CT scan, which can be obtained at reasonable cost from a number of facilities (or in the future from the rapidly improving technology of desktop CT scanning) and a few photographs under known illumination, which takes only a few minutes with a camera and a mirror sphere. The resulting models are volumetric in nature, and physically based, which makes them easier to edit than image-based data. It is easy to adjust color, glossiness, opacity, and material thickness by scaling parameters of the volume geometry; and a range of more fundamental changes to the material's structure can be made by editing the volume data.

This work has demonstrated the usefulness of the CT modeling approach for textiles, but the approach does have some limitations. Particularly, it requires that changes in optical properties correlate with changes in density, and this requirement could limit the kinds of materials that can be captured using this imaging modality. Further, the scanner can only image small samples, less than a centimeter across, at the resolution needed to produce clear fiber orientation maps. Thick materials that do not fit fully in the volume (e.g., materials with very long flyaway fibers) cannot be handled well. Some unusual materials, such as metallic fibers, may be problematic for CT because of limited dynamic range. Also, texture content at larger scales will be missed. These problems will decrease as CT scanners improve in resolution and dynamic range. CT is very well suited to textiles, and it remains to be seen what other materials it performs well for, and how other volume imaging methods work in this technique. Further, materials with differently colored yarns cannot be currently captured by our method.

There are many areas of future work. This work was done using extremely

small samples. With larger ones, which should be possible as CT technology improves and becomes more accessible, better texture could be produced.

To improve accuracy, more photographs under varying conditions can be used, allowing more parameters (for instance, more complex phase functions) to be fit. One possibility is to replace our optimization algorithm (which performs the binary search) with stochastic gradient descent methods (such as the one introduced by Gkioulekas et al. [31]) for solving a more general inverse volume rendering problem. Unlike [31] which focuses on isotropic media, we will need to derive unbiased gradient estimations with respect to the micro-flake parameters.

Ultimately, this method can be extended to work for a wide range of types of materials whose appearance is difficult to capture using surface models.

CHAPTER 5

STRUCTURE-AWARE SYNTHESIS FOR PREDICTIVE WOVEN FABRIC APPEARANCE

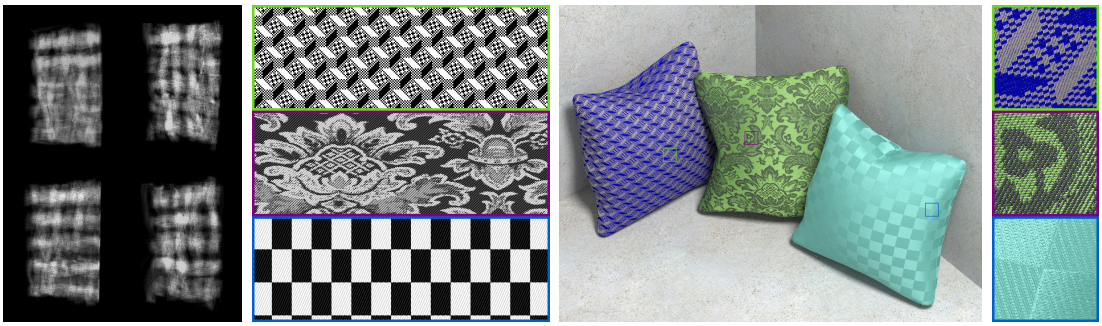
This chapter describes the second component of our main pipeline (Figure 1.3): a structure-aware synthesis framework that creates micron-resolution fabric models with complicated, user-specified designs.

Our approach starts with creating a database of example fabric models with elementary patterns. Given a new design specified by the user, our method creates a new volume by copying data from the exemplars at each yarn crossing. The resulting volumetric models can produce highly realistic renderings at both large and small scales. This work was originally published at ACM SIGGRAPH 2012 [112] and has been used by textile designers at Rhode Island School of Design to build a visualization system that greatly accelerates the design process. Figure 5.1-c has been selected as the front cover of the SIGGRAPH 2012 conference proceedings.

5.1 Introduction

Woven fabrics are common in everyday life and display highly varied appearance, with very fine detail and subtle directional effects that are created by the interplay of geometric structure with fiber properties. Capturing these effects in predictive renderings for arbitrary woven fabrics is a major challenge.

Realistic cloth is important for graphics applications from entertainment to apparel rendering, and it is also important in textile design. Textile designers use software such as Pointcarré [81] to design weave patterns for fabrics, and then drive industrial looms using the output. However, these packages do not



(a) (b) (c) (d)

Figure 5.1: We synthesize volumetric appearance models of fabrics with **complex designs** using a small set of exemplars: (a) density information of exemplars obtained using micro CT imaging; (b) fabric designs specified by weave patterns; (c) rendered results using synthesized volume data; (d) insets showing details: see, for example, blue yarns (top inset) hidden beneath the gray ones that are visible through the gaps.

provide realistic previsualization of the design before fabrication, thus forcing designers to fabricate "in the dark". Since loom time can be expensive and difficult to schedule, refining a design requires slow and costly iteration. By providing the ability to predictively preview the appearance of fabric designs before they are woven, we can minimize the need for test weaving, saving considerable time, raw materials, and cost.

Volume modeling and rendering techniques [48, 79] have recently been quite successful in capturing the diverse appearance of fabrics [108, 43, 111]. In particular, the technique presented in Chapter 4 builds volumetric models for fabrics using yarn and fiber geometry information from micro computed tomography (CT) imaging and optical information from a photograph. For materials with simple repeating structure and a single type of yarn, the volume data is tiled to produce large areas of fabric, with highly realistic results.

The key to this approach is to accurately model the geometric structure of the surface layer of the cloth, from which the many appearance phenomena of

different fabric types emerge automatically. However, this method is limited to materials containing only a single type of yarn, and it can only reproduce the exact material that was scanned. But real fabrics are complex—including intricate weave patterns, large scale designs, and multiple yarn types for warp and weft, each with its own reflectance; ideally, flexibility to render such fabrics is desired. Further, in design applications, the real usefulness of rendering comes from predicting the appearance of new fabrics that have not been scanned.

In this chapter, we present a new technique to create volumetric models of fabrics with complex, spatially varying structures and to predict the appearance of specific weave patterns. A small set of exemplars obtained from micro CT scans of simple fabrics are used to model new fabrics defined by 2D binary images representing each fabric's weave structure. Our structure-aware volume synthesis algorithm efficiently copies regions of the exemplars to assemble a volumetric model that matches the fabric's structure, without visible seams or periodic patterns. An additional edge fixing step is introduced to further improve the quality of the synthesized volumes.

We demonstrate our technique with synthesized results for a range of common structures that convey very different appearance. In some cases we use real weave patterns and compare to photographs of samples woven from the same patterns, and in other cases we use patterns generated to achieve a particular rendered appearance. This work can have impact on graphics applications in entertainment, e-commerce and apparel visualization, and textile design.

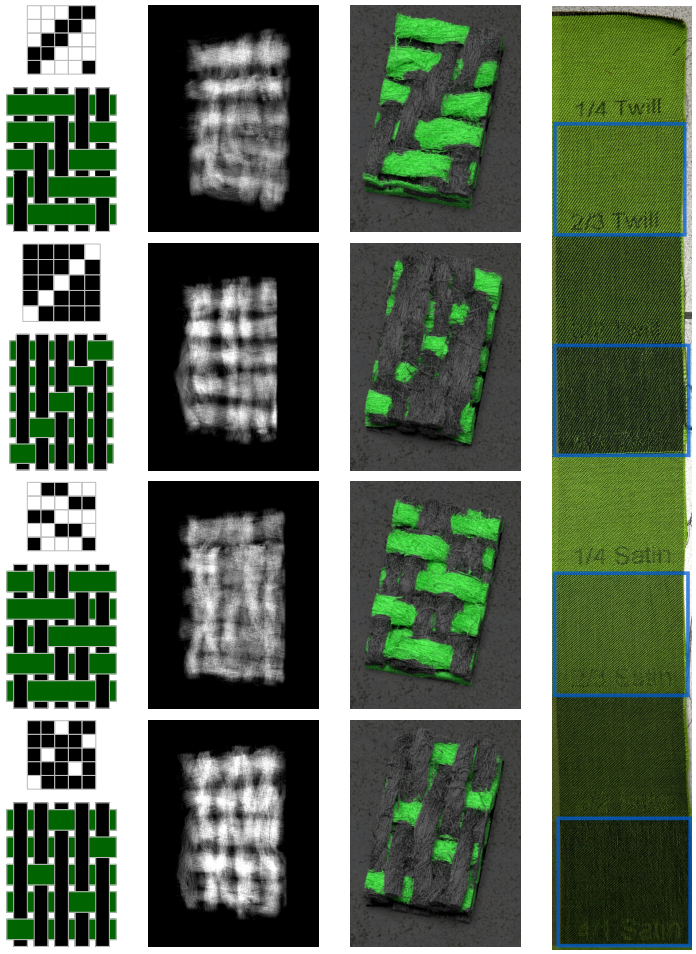


Figure 5.2: **Example weave patterns:** twill (top two rows), satin (bottom two rows); (a) weave patterns and the corresponding 2D illustrations where warps and wefts are respectively drawn in black and green; (b) CT data of fabric samples with the same weave patterns; (c) colored visualizations of the CT data; (d) a photograph of our example fabric in which the four examples used in this figure are marked with blue rectangles.

5.2 Background

Weaving is a process of interlacing two perpendicular sets of yarns, called the *warp* and the *weft*, to form a fabric. During the weaving process, warp yarns are fixed to the loom while weft yarns are inserted crossways, and different subsets of warp yarns are raised above or lowered below each inserted weft yarn so

that the yarns become interlaced and the fabric holds together into a sheet by friction. In this work, we follow the convention that warps go vertically and wefts horizontally.

Depending on the pattern in which warps are raised, fabrics with very different appearance and mechanical properties are produced. The pattern is described very simply using a binary image called a *weave pattern*, with the number of columns and rows equal to the number of warps and wefts in the cloth; a black pixel means the warp is above the weft at the corresponding yarn crossing, while white means it is below. Depending on the mechanics of the loom, only certain kinds of patterns may be achievable, but in the most general case of Jacquard looms, every yarn crossing is individually controlled by a computer. Figure 5.2 shows four different weave patterns from the *twill* and *satin* families that we use as exemplars.

Twill is one of the most common weave patterns, in which each row is shifted by one yarn from the previous row. As shown in the top row of Figure 5.2, fabrics created with twills convey characteristic diagonal lines. Simple twills are denoted "m/n twill" meaning a warp goes over m wefts, then under n wefts, then repeats. A twill pattern that repeats every k yarns is called a "k-end twill" or just "k-twill." The two 5-twill patterns shown in Figure 5.2 are $2/3$ twill and $4/1$ twill.

As opposed to twill, satin weaves shift each row by more than one yarn, with the aim of creating a more distributed pattern that does not call attention to the repeating structure. Satins build smooth surfaces and can be used for creating fabrics with glossy appearance. Satins are named in the same way as twills; the bottom rows of Figure 5.2 show a $2/3$ and a $4/1$ satin. Much larger satin patterns are often used with fine yarns when a glossy surface is desired.

In Jacquard fabrics with many-colored graphic patterns, the variety of available colors can be increased by using a multi-layer weave, so that some yarns can be hidden at the back of the cloth in areas where their color is not desired on the surface. The structures of double- and triple-cloth fabrics can be intricate, but for our purposes we are primarily interested in the yarns visible on the surface. Therefore in this work we treat cloth as if it were single-layered, with yarns that can change color along their length. In reality, of course, yarns do not change color but rather are substituted with other yarns that were previously hidden on the back, but the errors induced by this simplification are negligible.

5.3 Overview

The goal of our cloth modeling process is to produce volume models of woven materials, suitable for realistic close-up renderings, from two inputs: a description of the material to be simulated, and a few examples of similar but simpler fabrics. Our system accomplishes this in two phases. In the first phase, which only needs to be done once for a whole class of materials, CT scans of the example fabrics are used to build *exemplars* that contain all the information needed to synthesize large areas of complex fabrics. In the second phase, which is done once per material to be simulated, the exemplars are used in a new structure-aware volumetric texture synthesis method to synthesize a volume model according to the colors and weave pattern of the target material.

Exemplar creation phase. The purpose of the exemplar creation phase is to turn raw volume data into a database that can be used to synthesize volume models of a range of fabrics that are made from materials similar to the example

materials, but have different structure. Normally the example materials are a set of simple weaves made using particular types of yarns.

As input we assume volume data showing the geometric structure of the input fabric. While other data sources, such as magnetic resonance imaging (MRI), could be used, here we focus on volume datasets that come from CT scans. Thus, our input is the raw volume containing density information on a fine voxel grid covering a small patch of a fabric. High resolution scans are required to resolve fiber orientation and flyaway fibers, so each scan observes an area on the order of 5mm across, which, after cropping, typically produces exemplars with about 6×9 yarn crossings.

A processing pipeline takes this data and produces output by denoising the input density data, automatically tracking yarns in the data to detect the yarn trajectories, segmenting the voxels to match them to the appropriate yarns, and then automatically detecting the pattern of yarn crossings.

Each exemplar in the resulting database includes a voxel grid (containing density, fiber orientation, and yarn ID) and a small binary image representing the weave pattern. Section 5.5 gives details.

Synthesis phase. The input describing a new fabric to be simulated includes a 2D binary array giving the weave pattern for the whole cloth, and 2D arrays specifying the type of warp and weft yarn present at each yarn crossing. The synthesis phase, detailed in Section 5.4, creates an output volume that respects the input specification while displaying local structure and details that match the exemplars.

How the fabric specification will be created depends on the application.

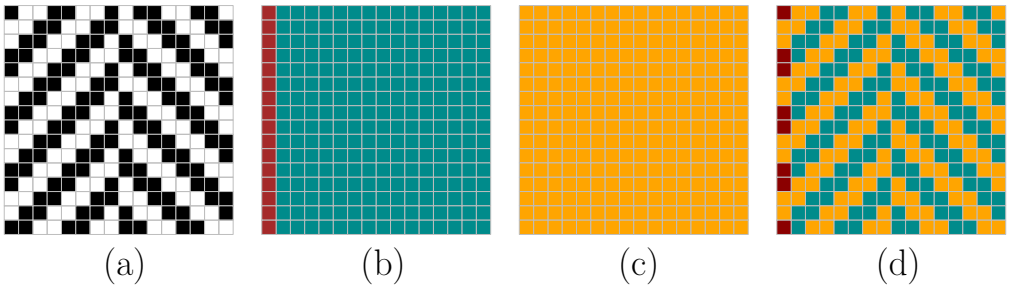


Figure 5.3: **Inputs to our algorithm:** (a) shows the weave pattern; (b) and (c) show warp and weft ID maps encoded in colors, indicating that all but the left-most warp share the same optical properties while all wefts are identical; (d) illustrates the visible yarn ID at each crossing.

When predicting the appearance of a new fabric as part of the textile design process, this description can be extracted directly from the actual design, using the data that would be sent to the loom to make the fabric. In a graphics context, the weave and color pattern can be computed from a posterized image by a very simple process, since the constraints of actual weaving do not need to be observed, as described in Section 5.4.2.

5.4 Structure-aware synthesis

Given appropriate exemplars, structure-aware synthesis produces a detailed model of a fabric from a description of the required design.

5.4.1 Input specification

The input to our algorithm (Figure 5.3) consists of three components:

1. A binary image W representing the weave pattern, where each binary value represents a yarn crossing.

2. A 2D array specifying the warp and weft types, represented with IDs, for every yarn crossing.
3. Color and gloss information for each type of yarn.

As discussed in Section 5.2, we model multilayer weaves by allowing a single warp or weft to change color along its length. This also provides flexibility in graphics applications where the cloth need not be manufacturable, because the color constraints of the actual weaving process can be discarded if desired. For this reason the yarn color arrays are two-dimensional rather than one-dimensional. In reality, there are usually no more than ten different kinds of yarns in a fabric.

5.4.2 Input data creation

The input above can be created in several ways. A textile designer would normally use design software such as Pointcarré [81] to design a new material, and the software can simply output the required binary weave pattern and yarn color information.

For applications where constraints of producing actual cloth are less of a concern, the following simple method mimics the design process. Begin with a posterized image I and a set of q different elementary weave patterns $V_1, V_2, \ldots, V_q$ (repeated until they match the size of I) associated with warp and weft ID maps. Note that the set of elementary patterns can be arbitrary and does not need to be contained in our exemplar database. Let the set of discrete colors in I be $\mathcal{P}$. The user then assigns a weave pattern to each of the colors by specifying a mapping $p : \mathcal{P} \mapsto \{1, 2, \ldots, q\}$. To produce the patterned cloth, take each pixel from that

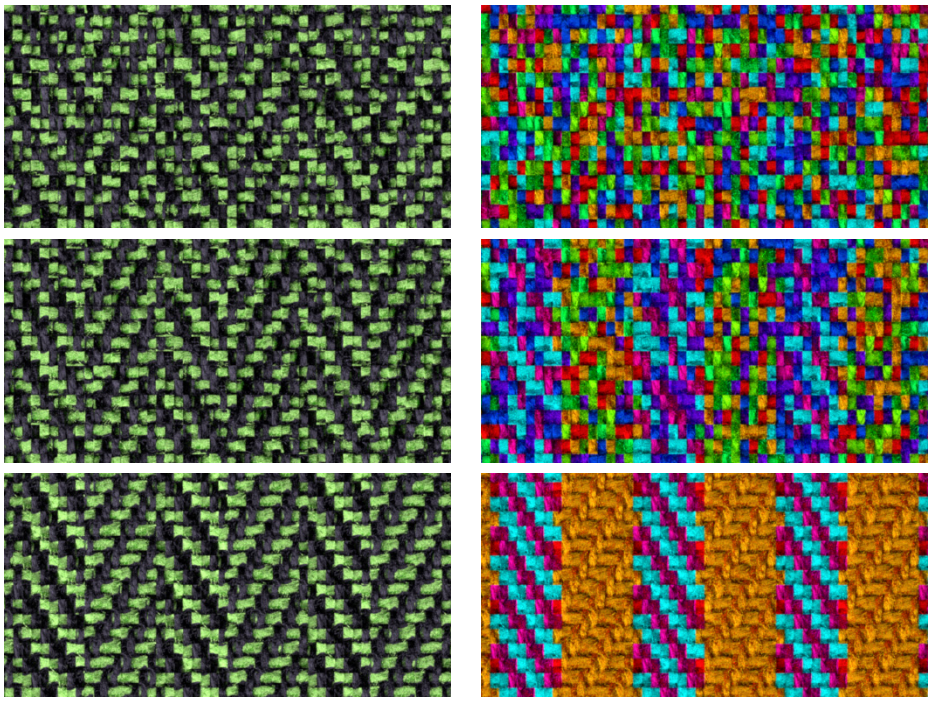


Figure 5.4: **Synthesized results** using (top) naive algorithm, (middle) greedy algorithm, (bottom) our approach: the left column shows renderings using synthesized models; the right column shows from which exemplar each block copies its content (encoded in false colors).

of one of the elementary weave patterns as follows: $W(x, y) = V_{p(I(x,y))}(x, y)$.

5.4.3 Synthesis at the yarn level: the problem

The goal of our synthesis process is to generate a large volume matching the given weave pattern. At the *voxel level*, the problem is to solve for the value of each voxel of output. Since the total number of voxels can be very large (a $1\text{m} \times 1\text{m} \times 2\text{mm}$ cloth sampled at $5\mu\text{m}$ resolution has 1.6×10^{13} voxels), computing or even storing the solution is costly and must be avoided. At the *yarn level*, the algorithm instead considers one pixel in the weave pattern (which represents a warp-weft yarn crossing) at a time, and "copies" the corresponding volume data, which we call a *block*, by referencing a rectangular box in an exemplar volume.

This is much more tractable, so given the costs involved, we solve the problem at the yarn level, and then apply a post-process to effectively adjust the data at the voxel level to get a high quality synthesized result.

The core problem of the yarn-level synthesis process is to locate a block in the exemplar volumes for each pixel in the weave pattern to copy its volume contents from. Let A be a weave pattern associated with a volume $\tilde{A}$, and let $\tilde{A}(i,j)$ denote the block in $\tilde{A}$ corresponding to pixel $A(i,j)$. Then the yarn-level synthesis problem can be formulated as follows: given a set of k example weave patterns $\{S_1, S_2, \ldots, S_k\}$ associated with k exemplar volumes and a target pattern W, determine an *assignment function* $c : \mathbb{N}^2 \mapsto \mathbb{N}^3$ where $c(i,j) = (u,x,y)$ indicates that $\tilde{W}(i,j)$ copies its data from $\tilde{S}_u(x,y)$. Note that c can be implemented simply as a 2D array.

One possibility is to randomly copy blocks that have the desired yarn (warp or weft) on the top, by assigning $c(i,j)$ a random triple (u,x,y) under the constraint that $W(i,j) = S_u(x,y)$. Unfortunately, this works poorly, as shown in the top row of Figure 5.4, since there is no consistency across block boundaries. The shape of the yarn passing through a given block is affected strongly by whether it passes under or over the next yarn, and for this reason it is critical to ensure that the binary values of the four neighboring pixels match when selecting an exemplar block to copy.

Thus our method follows three principles when selecting an exemplar block for each output block, enforcing them in priority order:

1. *Correctness.* The correct yarn (warp or weft) must be on top.
2. *Consistency.* The four neighbors in the desired weave pattern should match the neighbors in the exemplar's weave pattern.

3. *Continuity.* Choices that copy neighboring blocks in the exemplar into neighboring blocks in the output are preferred.

Next we define the term *consistency*. For every element $(i, j) \to (u, x, y)$ of c, which we call an *assignment*, consider the four neighbors of $W(i, j)$ and $S_u(x, y)$. If all these neighbors match in binary values, we say the mapping is *consistent*. Unfortunately, it is not always possible to find c such that all assignments are consistent. Thus the problem can be described as an optimization: find an assignment function such that the total number of matches is maximized. Figure 5.5 shows an example where the block in $\tilde{S}_3$ maximizes the number of matching neighbors.

Note that to maximize the total number of matches, the choice for one assignment is independent of that for another. This fact suggests a greedy algorithm: for each pixel $W(i, j)$, select the triple with the maximum number of matches. Although this simple algorithm can generate much better results (see the middle row of Figure 5.4), it does not provide local continuity since the algorithm does not know how to break a tie when there are multiple candidates with the same number of matches. This is unfortunately a very common situation over uniform regions of a fabric.

To tackle this problem, we introduce a "continuity" term as follows. Given an assignment function c, let the *continuity* at (i, j) be the total number of immediate neighbors (i', j') satisfying the following conditions: let $c(i, j) = (u_0, x_0, y_0)$ and $c(i', j') = (u_1, x_1, y_1)$, then $u_0 = u_1$ and

$$(x_1, y_1) - (x_0, y_0) = (i', j') - (i, j).$$

Our problem is to find an assignment maximizing the total consistency, using the total continuity to break any ties.

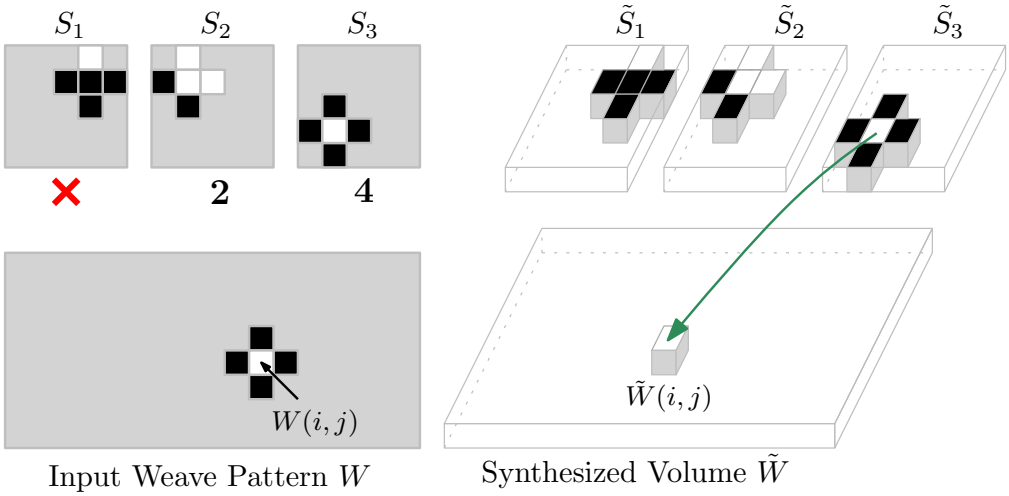


Figure 5.5: The block in $\tilde{S}_1$ is not a valid candidate since it does not satisfy the correctness constraint; the blocks in $\tilde{S}_2$ and $\tilde{S}_3$ satisfy the constraint and respectively have 2 and 4 matching neighbors.

5.4.4 Synthesis at the yarn level: our algorithm

While matching consistency can be done using a greedy algorithm, maximizing the total continuity on a 2D grid is in general a very hard combinatorial optimization problem. Fortunately, the 1D version of this problem, where continuity is defined by considering the two immediate neighbors for each yarn crossing, can be solved efficiently. And our experiments indicate that solving the 1D problem for every column (along the warps) is a good approximation of the 2D problem when handling weave patterns.

For convenience, we associate a unique index to each triple (u, x, y) used by the assignment function c. Beyond this point, we assume that $c(i, j)$ returns a single integer instead of a triple.

For a column y_0 in the weave pattern, let $f(i, t)$ denote the maximal total continuity for the first i rows in this column under the constraint that $c(i, y_0) = t$. And $f(i, t)$ is defined only when t maximizes consistency at $W(i, y_0)$. To compute

$f(i,t)$, we can solve recursive sub-problems over the first $(i-1)$ rows, namely computing $f(i-1,t')$ for all feasible t' values, and then pick the best one to form $f(i,t)$ by computing

$$f(i,t) = \max_{t'}\{f(i-1,t') + \text{gain}(t',t)\} \tag{5.1}$$

where $\text{gain}(t',t)$ captures the 1D continuity and equals 1 if t' and t come from the same column of one exemplar volume and t' lies next to t and 0 otherwise. The base case of this recursion is $f(0,t) = 0$ for all t.

Note that the total number of states is polynomial, and this optimization problem can be solved efficiently using dynamic programming. Figure 5.6 illustrates the process of computing $f(i,t)$ by enumerating t' values.

Algorithm complexity. Given a target weave pattern of size $M \times N$, and k example weave patterns each of size $m \times m$, the dynamic programming algorithm runs in $O(km^2MN)$ time. In our experiments, $m = 5$, $k = 8$, and M, N can be as large as several thousands.

Algorithm optimization. Many basic weave patterns are translationally symmetric: each column is a translated version of the previous one (with wrapping around the edges). It therefore suffices to consider just one column, reducing the time to $O(kmMN)$.

Randomization. The above optimization could result in a loss in variation by not considering the other $m-1$ columns. Further, a side effect of maximizing local continuity is that it may create periodic patterns. To solve this problem, we randomly shift each block based on translational symmetries of the corresponding exemplar. And we shift every $\hat{m} \times \hat{m}$ block by the same amount to avoid

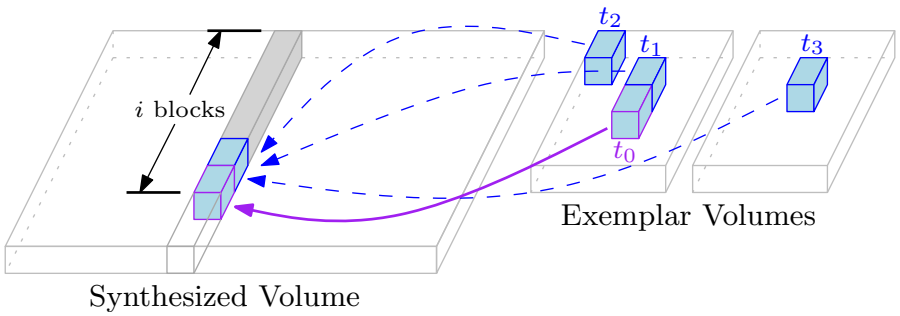


Figure 5.6: **The dynamic programming process:** computing $f(i, t_0)$ by enumerating different possible t' values using Equation 5.1. For example, here we have $\text{gain}(t_1, t_0) = 1$ whereas $\text{gain}(t_2, t_0) = \text{gain}(t_3, t_0) = 0$.

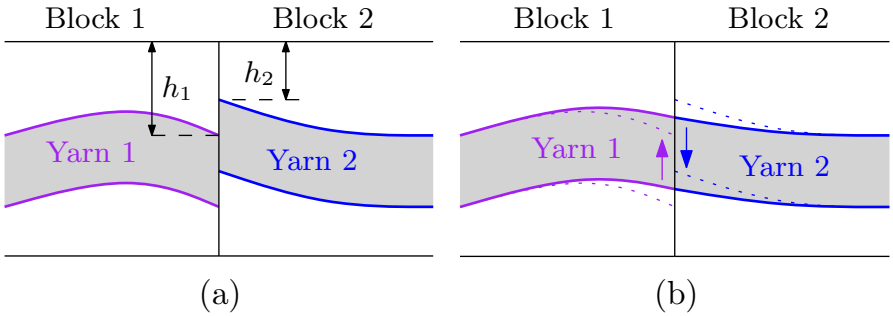


Figure 5.7: Fixing the edges by moving stacks of voxels.

destroying the continuity obtained by solving the optimization problem. Since $m = 5$ in our experiments, we have picked $\hat{m} = 3$.

Discussion. The 1D dynamic programming can be also performed along the weft direction or even alternating between the two directions. In our experiments, these schemes all produced very similar results. This is because the input weave pattern itself has strong 2D structure, and the 2D neighborhoods greatly restrict the set of candidate blocks at each yarn crossing. As shown in Figure 5.4, our algorithm provides good continuity in both directions.

5.4.5 Edge fixing

Recall that each block represents one warp-weft yarn crossing. When two adjacent blocks copy their contents from disconnected blocks, visible seams may be created, as shown in Figure 5.9. We introduce an efficient method to fix the seams by shifting entire stacks of voxels along the Z direction.

Fixing a single block. Figure 5.7a illustrates the 2D case where both blocks contain a single yarn and there is a seam on the edge between them. Assume that the right end of yarn 1 and the left end of yarn 2 have depths h_1 and h_2, respectively. Then the seam can be fixed by shifting up yarn 1 by $\Delta h := (h_2 - h_1)/2$ on the right, and yarn 2 by $(-\Delta h)$ on the left.

For the 3D case, the desired amount of shifting is defined along the 2D boundary of each block. However, only shifting the boundary stacks will create new seams. We also need to update the inner stacks so that the entire adjustment is smooth. We do this smooth adjustment of depths by solving a 2D discrete Poisson equation.

Note that we focus on matching the top-most yarns. In the case of a multi-layered fabric, this may cause errors (seams) for the yarns underneath. Fortunately, they can be safely ignored since we cannot see those yarns directly and the amount of shifting involved by this process is generally much smaller than the radius of a single yarn.

Fixing all blocks. Although fixing one block is relatively easy, given that the total number of blocks is in millions, the time and storage needed to compute and store the solutions become prohibitive. However, the problem can be solved

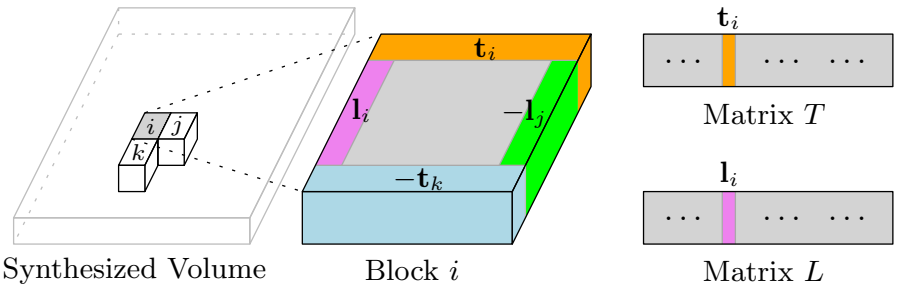


Figure 5.8: **Edge fixing:** constructing matrices T and L. The structure of block i is shown in the middle, and assume that the blocks to its right and bottom are respectively block j and k.

efficiently if all blocks have identical resolutions $b_1 \times b_2 \times b_3$, which is normally the case if the fabric samples are scanned using the same resolution.

For every block, say block i, let the boundary condition defined on its upper edge be $\mathbf{t}_i$ and that on its left edge be $\mathbf{l}_i$. Stack all $\mathbf{t}$ and $\mathbf{l}$ vectors as columns to form two matrices T and L with dimensions $b_1 \times MN$ and $(b_2 - 2) \times MN$, respectively. The boundary conditions defined on a block's right and bottom edges can be represented using those on its right neighbor's left edge and bottom neighbor's top edge, so they do not need to be stored. Figure 5.8 illustrates this process.

Next, we compute rank-r approximations for the matrices T and L: $T \approx B_T \times C_T$; $L \approx B_L \times C_L$ where B_T, B_L respectively have dimensions $b_1 \times r$, $(b_2 - 2) \times r$, while C_T, C_L are both $r \times MN$. In our experiments, we tried multiple r values and found that $r = 15$ works well. Then we solve the $2r$ Discrete Poisson equations whose boundary conditions are given by each column of B_T and B_L (and setting all other edges to 0). We store the solutions $\mathbf{s}_t^T$ and $\mathbf{s}_t^L$ for $t = 1, 2, \ldots, r$. It follows that the solution of the Poisson equation defined on each block is (approximately) a linear combination of the stored values, since the solution of a Poisson equation is a linear function of the boundary condition.

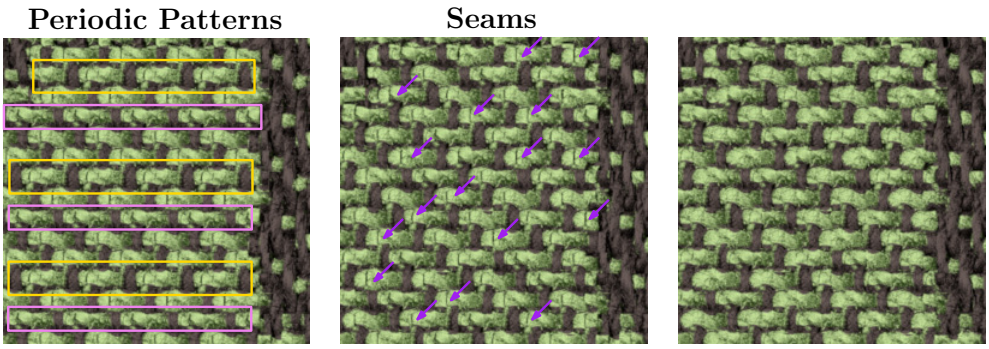


Figure 5.9: **Randomization and edge fixing:** (left) maximizing consistency and continuity without randomization results in periodic patterns; (center) introducing randomization removes such patterns; (right) edge fixing significantly improves the seams.

For block i, assume its right and bottom neighbors are block j and k, respectively. Then the solution $\mathbf{s}_i$ at block i equals

$$\sum_{t=1}^{r} \left(C_T(t,i)\, \mathbf{s}_t^T + C_L(t,i)\, \mathbf{s}_t^L - C_T(t,k)\, \tilde{\mathbf{s}}_t^T - C_L(t,j)\, \tilde{\mathbf{s}}_t^L \right)$$

where $\tilde{\mathbf{s}}_t^T$ and $\tilde{\mathbf{s}}_t^L$ are the vertically and horizontally mirrored copies of $\mathbf{s}_t^T$ and $\mathbf{s}_t^L$, respectively.

In our implementation, we precompute C_T, C_L, $\mathbf{s}_t^T$, $\mathbf{s}_t^L$ and obtain $\mathbf{s}_i$ at runtime. In our experiments, storing this information takes roughly 150 MB of space. Finally, when the voxel contents at location $\mathbf{p} = (p_x, p_y, p_z)$ need to be fetched, we adjust p_z by $\mathbf{s}_i(p_x, p_y)$ before performing the volume lookup.

We have shown how to synthesize the volume data using several exemplars. Next we will provide the details on creating those exemplars.

5.5 Exemplar creation

In this section we describe our pipeline of CT image processing. The goal is to create a set of exemplar volumes in which each voxel contains three parameters:

material density, local fiber orientation, and a yarn ID. The material density and fiber orientation can be passed directly to the microflake model [43], and the ID can be used to tell the type of yarn (warp or weft) and then obtain the corresponding optical information from the input images. This information combined yields a complete volumetric appearance model which can then be rendered.

This stage starts with a basic processing step following [111] which takes raw CT data and produces density and orientation information for each voxel. Next we compute per-voxel yarn IDs and regularize the exemplar volumes.

5.5.1 Yarn tracking

To obtain the yarn ID for each voxel, we reconstruct the center curve, a discrete line strip, for each yarn in the volume; this process is called *yarn tracking*.

Tracking. Given a yarn passing 3D point $\mathbf{p}$, we can track it by iteratively computing the tangent direction $\mathbf{t}$ at the current location and moving in that direction by a small step d. The tangent direction $\mathbf{t}$ can be computed by averaging the local fiber orientation over a small region around $\mathbf{p}$:

$$\mathbf{t} = \text{normalize}\left(\sum_{\mathbf{v}\in V_{\mathbf{p},\mathbf{t}_0}} \omega_f(\mathbf{v})\right)$$

where $\mathbf{t}_0$ is the tangent direction computed in the previous iteration, $\omega_f(\mathbf{v})$ is the local fiber orientation at $\mathbf{v}$, and $V_{\mathbf{p},\mathbf{t}_0}$ is a volume around $\mathbf{p}$ defined by

$$V_{\mathbf{p},\mathbf{t}_0} = \left\{\mathbf{v} \in \mathbb{R}^3 \ : \ \|\mathbf{v}-\mathbf{p}\|_2 \le R \ \text{ and } \ |\omega_f(\mathbf{v})\cdot\mathbf{t}_0| \ge c_0\right\}$$

where d and c_0 are constants representing the size of a yarn (in voxels) and its degree of deformation. In our experiments, $d = 15$, $c_0 = 0.7$, and R respectively

equals 20 and 30 when tracking warps and wefts.

If the input orientation field contains too much noise, the estimated $\mathbf{t}_0$ will be inaccurate, which may cause the tracking step to fail. We therefore need to ensure that the CT scans have not only sufficient resolution but also acceptable signal-to-noise ratio, which can be ensured by using longer exposures.

Endpoint detection. To start the tracking process, an endpoint of every yarn is needed. We detect the endpoints automatically using K-means clustering (with the total number of clusters as user input).

We assume that the warps approximately go vertically (along the Y-axis) while the wefts run horizontally. This can be easily satisfied by roughly aligning the samples during the scanning process. Also, we assume that the user knows the number of warps and wefts in the scanned volume.

To detect the warp centers on a 2D slice perpendicular to the Y-axis, we perform a K-means clustering among those voxels on the slice whose fiber orientations are close enough to the Y direction; the center of each cluster indicates the warp centers. Similarly, the centers of the wefts can be detected by running the same algorithm for any slice perpendicular to the X-axis.

Note that the computed yarn centers can be unreliable: noise and voxel orientation errors can result in poor accuracy. We therefore run this process for every slice along each axis and pick the one which minimizes the maximal cluster radius.

Correction. Tracking a yarn by simply following the tangent direction causes $\mathbf{p}$ to leave the yarn in highly curved regions. Therefore, we introduce a correction

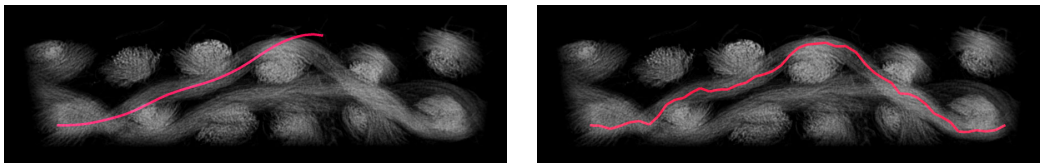


Figure 5.10: **Center correction:** (left) without the correction, the tracking that starts from the left fails due to the yarn center leaving the volume; (right) with the correction, the tracking process becomes more robust.

step after each tracking iteration. Every time $\mathbf{p}$ is updated with $(\mathbf{p} + d \cdot \mathbf{t})$, we iteratively move $\mathbf{p}$ to the center of mass of a small volume around it:

$$\mathbf{p} \leftarrow \frac{\sum_{\mathbf{v} \in S_{\mathbf{p},\mathbf{t}}} \rho(\mathbf{v}) \cdot \mathbf{v}}{\sum_{\mathbf{v} \in S_{\mathbf{p},\mathbf{t}}} \rho(\mathbf{v})}$$

where $\rho(\mathbf{v})$ denotes the material density at $\mathbf{v}$, $S_{\mathbf{p},\mathbf{t}}$ is a 2D region surrounding $\mathbf{p}$ on the slice which is perpendicular to the main direction of the yarn and contains $\mathbf{p}$. In our experiments, we made $S_{\mathbf{p},\mathbf{t}}$ elliptical to provide more freedom for $\mathbf{p}$ to move along the Z-direction while preventing it from accidentally jumping to a neighboring yarn. As shown in Figure 5.10, the correction step significantly stabilizes the tracking process.

Discussion. Shinohara et al. [90] proposed a similar yarn tracking approach. However, their method requires the user to enter the endpoint for each yarn and does not have the correction step which has proven crucial for tracking the yarns in our experiments.

5.5.2 Weave pattern detection

With the tracked yarns in hand, we would now like to infer their associated weave pattern. For each yarn crossing, this entails determining whether the weft passes above or below the warp. Mathematically, this property is captured by

the *linking number* of the yarn curves [86]. Reminiscent of the winding number in two dimensions, the linking number counts the signed number of times a space curve wraps around another. Given parameterizations $\mathbf{p}_1(t_1)$ and $\mathbf{p}_2(t_2) : [0,1] \rightarrow \mathbb{R}^3$ of a warp and weft, respectively, we numerically compute their linking number using the Gaussian linking integral

$$L_{1\leftrightarrow 2} = \frac{1}{4\pi} \int_{[0,1]^2} \left\langle \frac{\mathbf{p}_1(t_1) - \mathbf{p}_2(t_2)}{\|\mathbf{p}_1(t_1) - \mathbf{p}_2(t_2)\|_2^3}, \frac{\partial \mathbf{p}_1}{\partial t_1} \times \frac{\partial \mathbf{p}_2}{\partial t_2} \right\rangle \mathrm{d}t_1 \mathrm{d}t_2$$

Assuming that the warps and wefts are parameterized so that the projection of their tangents into the plane of the weave pattern forms a positively oriented basis of $\mathbb{R}^2$, we assign the value 0 to this yarn crossing if $L_{1\leftrightarrow 2} > 0$ and 1 otherwise.

Using this technique we can automatically detect weave patterns for single-layered fabrics. For those with multiple layers, we must manually reason about the layered structure to find the top-most yarn for each weave grid location before computing the linking number.

5.5.3 Voxel segmentation

Based on the tracked yarns, we can tell which yarn each voxel belongs to. This information is needed in the later steps of the pipeline. For each voxel, we assign it to the yarn whose center curve minimizes the distance to the voxel. Note that because the warp and weft yarns have different radii and stiffness, this simple approach can cause small errors at the yarn crossings. Those errors, however, are hardly visible in rendered results since they are hidden beneath the surface.

5.5.4 Volume alignment

Since the samples are not perfectly registered during the scanning process, they need to be aligned before being used for synthesis.

We solve for a global rotation around the Z-axis for each scanned dataset. For the yarn i, let Y_i be the set of voxels it contains. We perform PCA on $\{(v_x, v_y) : \mathbf{v} = (v_x, v_y, v_z) \in Y_i\}$ to detect the principal direction of the yarn. Let ω_0 and ω_1 be the average principal directions of the warps and the wefts respectively. We rotate the whole volume around the Z-axis so that $(\omega_0 + \omega_1)/2$ matches the diagonal line $y = x$.

5.5.5 Weaving grid registration

Given a volume with the associated weave pattern, we need to crop out the incomplete yarns on the boundary and make the remaining part aligned with the weaving grid. This sub-yarn level registration is crucial for the quality of synthesized data since our algorithm works at the yarn level and may copy incomplete parts of yarn crossings if the yarns are not centered in the blocks.

We first estimate the size (w, h) of the crop window by multiplying the number of complete yarns and their average sizes. Then the problem becomes that of finding a translation (x, y) such that the content in the crop window agrees with the weave pattern best.

Assume that the scanned volume has size $s_1 \times s_2 \times s_3$. We compute an $s_1 \times s_2$ binary image b in which $b(i, j)$ is set to 0 if the topmost non-blank voxel located at (i, j) is part of a warp yarn and 1 otherwise. For each grid in the window corresponding to one pixel in the weave pattern, we assign it a score that equals

the fraction of pixels in b that agree with the weave pattern value. The best crop window maximizing the total score of all grids can be computed in $O(s_1 \cdot s_2)$ time using a summed area table.

For our scanned data, $s_1 = s_2 = 1000$, $s_3 = 300$, $w = 575$, and $h = 350$. Using this configuration, each cropped volume contains 5×5 yarn crossings. Thus every block has the resolution $115 \times 70 \times 300$.

5.5.6 Summary

We create our exemplar database by taking the processed CT data, tracking the yarns, computing per-voxel yarn IDs, detecting the weave pattern, and regularizing the volume. Our pipeline requires a small amount of user input including the thresholds for tracking, the number of yarns for endpoint detection, yarn grouping for weave pattern detection, and the resolution of a single block for weaving grid registration. For creating each of our exemplars, the entire process (excluding the basic processing step from Chapter 4) runs in seconds.

5.6 Experimental results

We show two types of results to demonstrate our technique: comparisons of our synthesized results with real fabricated cloth samples, and new designs synthesized using our algorithm. Our renderings are generated using Monte Carlo volume path tracing implemented in the Mitsuba renderer [42].

First, to demonstrate the accuracy of our approach, we compare our results with real fabricated cloth samples. Several designs, including a "test blanket"

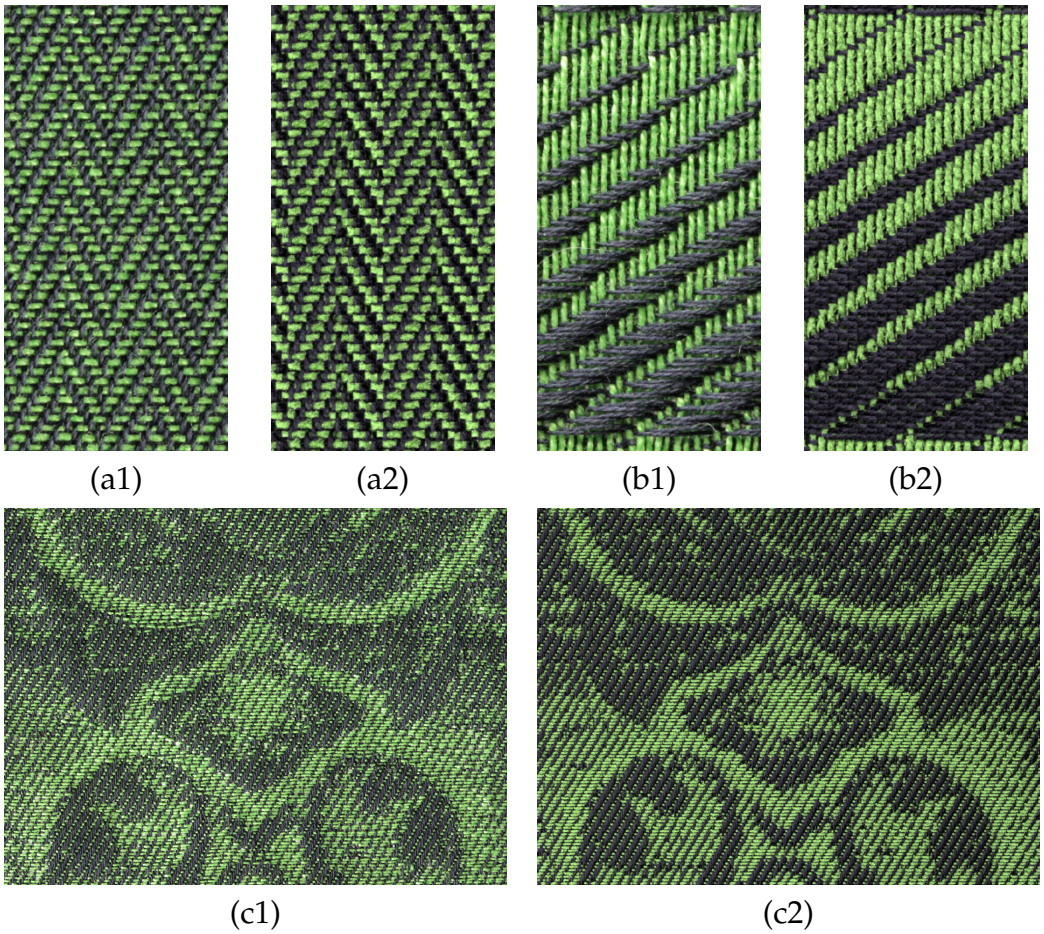


Figure 5.11: Comparisons between photographs of fabricated cloth samples (left) and rendered images with the synthesized data (right): (a) a Herringbone fabric; (b) a fabric containing all 9-twill patterns; (c) a Jacquard fabric (design courtesy of Brooks Hagan).

of 5-twill and 5-satin example patches (shown in Figure 5.2d), were woven on an industrial Jacquard loom (see Figure 5.12) at Rhode Island School of Design (RISD). Solving for the optical properties of multiple kinds of yarns in a fabric is beyond the scope of this work, so we manually picked colors for the black warp and green weft to obtain a rough match in appearance.

The results demonstrate our ability to correctly predict the structure and overall appearance of woven fabrics before they are fabricated, meaning that our methods are useful for textile designers who currently design "in the blind"

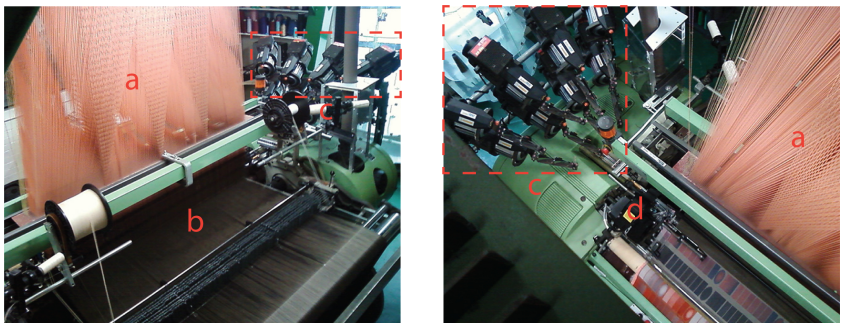


Figure 5.12: The industrial **Jacquard loom** at Rhode Island School of Design used to weave our samples: (a) *harnesses* used to lift the warps; (b) the warp yarns; (c) spools of multi-colored weft yarns; (d) the *shuttle* for carrying and inserting wefts.

without seeing any realistic preview of the cloth before fabrication.

We first pick two example weaves, Herringbone and a 9-twill pattern, that are not in our input set of exemplars, thus demonstrating the power of our approach to generalize to complex weaves. These are shown in Figure 5.11ab (in the 9-twill results, the fabric is rotated by 90° with the wefts going vertically). While our results are more regular than the photo, we successfully capture the key structure of both patterns. As shown in Figure 5.4, our algorithm copies big chunks of data if similar structures are contained in the database (the orange region) and synthesizes other regions from smaller pieces. Note that we are not allowed to rotate the exemplars, because the warp and weft yarns are dissimilar, or to mirror them, because that would reverse the twist of the yarns, destroying the consistency of local fiber orientations.

Next we synthesized a model using the weave pattern that was used to weave the Jacquard sample. Figure 5.11c shows a comparison between our synthesized result and the real sample.

Finally, Figure 5.13 show more results synthesized using our technique and mapped to arbitrary surfaces using shell mapping [83]. In each image the weave

pattern or the posterized image used to create the weave pattern is shown at the top left corner, and magnified insets appear at the bottom right.

All weave patterns used to synthesize these results have the resolution of 900×1500. Our synthesis algorithm runs in no more than 10 seconds to create each output volume containing 3.26×10^{12} effective voxels. Each rendered image has the resolution 2560×1440, and the rendering time varies from 15 to 40 minutes on a QSSC-S4R Intel Server with 32 logical cores.

Figure 5.13a shows a fabric created with a 96×96 wavy twill pattern. This input is very different from the patterns in our exemplar set; thus it is difficult to copy large pieces of continuous structure. Our approach does a good job of synthesizing the fabric with smooth shading across the surface.

In Figure 5.13b, we show a fabric containing alternating 1/15 and 15/1 satin blocks with all yarns assigned identical optical properties. This kind of same-color patterning is often used in fabrics for bed linens and draperies. Because the yarns go in perpendicular directions, the fabric has highly anisotropic appearance. By correctly synthesizing the structure of the yarn we are able to automatically capture this characteristic appearance. In addition, 3D structures created by the transitions between the two weave patterns can be easily observed even at a large scale.

Figure 5.13c shows a Jacquard cloth with 1/7 twill and 7/1 satin respectively forming the golden and the red area. The posterized image used to generate the pattern is shown on the upper-left. Again, the different structure of the two weaves results in distinctive anisotropic behavior that is captured well by our method.

Figure 5.13d shows a complex Jacquard cloth with two weft colors (white

and golden). The design is provided as an example in Pointcarré [81]. The synthesized result conveys a richly detailed surface structure.

5.7 Conclusion

We have demonstrated an approach to generate highly realistic volumetric models with spatially varying appearance and complex designs. Our approach takes advantage of micro CT imaging to measure highly detailed 3D structure and introduces a synthesis process to generalize the measured data to model fabrics with complex larger-scale structures. Our contributions include a robust pipeline for rapidly creating exemplars for fabrics, and a fast synthesis algorithm to create complex volumetric models. We have validated our synthesized results by comparing them to fabricated real samples.

We believe that this technique is very useful for both the computer graphics community and the textile design community. Using our exemplars, users can now create high quality fabric models with their own designs without having to write specialized code. And textile designers can use our approach to predictively visualize their designs without physically creating them.

There are multiple areas of future work. One limitation of our work is that our method can only synthesize models with a grid-like weave structure. We would like to extend our framework to support more structures (e.g., knitwork). For more automated, predictive rendering of existing fabrics, better appearance matching methods that solve for the optical properties of multiple kinds of yarns are needed. Finally, better optical models may be required to perfectly match the appearance of fabric samples.

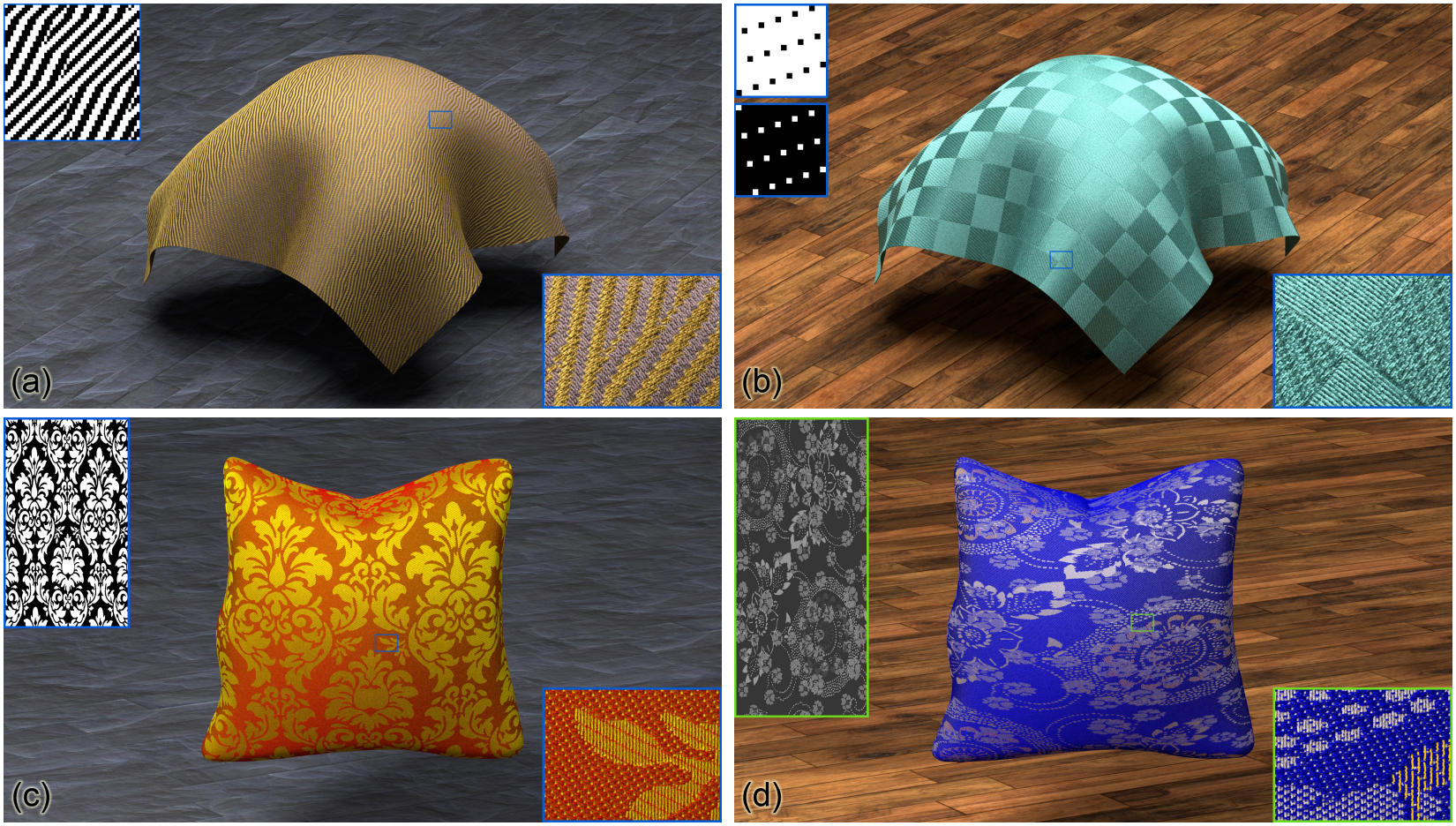


Figure 5.13: **Synthesized results with different weave patterns:** (a) a wavy twill; (b) a fabric composed using alternating blocks with 1/15 and 15/1 satin patterns; (c) and (d) Jacquard brocades mapped onto pillows. Top-left: weave patterns, Bottom-right: insets.

CHAPTER 6

MODULAR FLUX TRANSFER

In this chapter, we describe the third component of our main pipeline (Figure 1.3): a precomputation based algorithm for efficient rendering of high-resolution appearance models created in Chapters 4 and 5.

Since our models are structured, with repeated patterns approximated by tiling a small number of exemplar blocks, we precompute voxel-to-voxel, patch-to-patch, and patch-to-voxel flux transfer matrices for each of those blocks. Such precomputed information of an exemplar database can be reused to accelerate the rendering of any model synthesized from it. At render time, we introduce a modular flux transfer algorithm based on Monte Carlo Matrix Inversion (MCMI) to approximate higher-order scattering, which normally takes the majority of rendering time. This work originally appeared at ACM SIGGRAPH 2013 [110].

6.1 Introduction

High-quality rendering of complex materials increasingly uses volumetric data, such as the high-resolution micro-CT cloth models created using methods introduced in Chapters 4 and 5. However, rendering optically dense volumetric media is challenging for multiple reasons: their high resolution (with voxel sizes of a few microns), the complex occlusion in the medium, and the anisotropic phase functions that influence light scattering. Moreover, computation is dominated by multiple scattering within long light paths (5-100 scattering events), especially when the single-scattering albedo is high. This leads to an undesirable situation, where rendering brighter colored materials becomes substantially more expensive. To date, pure Monte Carlo path tracing has proven the only

(a) (b) (c)

Figure 6.1: We introduce a **modular flux transfer** (MFT) framework to approximate high-order scatterings in extremely complex volumes. (a) a scene containing a purple tablecloth with 2009×3300 yarn crossings represented by an anisotropic volume consisting of 1.06×10^{13} effective voxels; (b) a $5\times$ zoomed version of the left image, illustrating the complexity of the cloth volume; (c) a $25\times$ zoomed version.

reliable method for rendering such materials, but so far has been too slow for wide-spread use: tens to hundreds of core hours are required to produce the images in [111].

In this chapter, our goal is to significantly improve upon this situation; the approach we take is based on the following insights. First, complicated volumetric media like cloth often contain repetitive building blocks, such as yarn crossings. This suggests the possibility of precomputing light transport in these blocks, and *modularly* combining them into a complex volume: we are inspired by [62] who introduced a similar idea for approximating indirect lighting in blocked scenes. Second, in dense media with high albedos, such as white or bright-colored fabrics, high-order multiple scattering is the most expensive component and contributes significantly to overall appearance [72, 43]. This suggests that speedup is most likely to be achieved by accelerating the computation of the extremely expensive, but lower-frequency multiple scattering. A similar approach is common for subsurface scattering, but diffusion approximations do not easily apply to the complex volumetric media we are interested in, since they are often based on assumptions of isotropy, homogeneity, and flat boundaries.

The key challenge to a scalable solution is high dimensionality. In the most general case, light transport is a linear operator that maps emitted radiance into final radiance. The radiance on the boundary of a volume is a function of position and direction: a 4-dimensional light field. Worse yet, a precomputed approach seems to require the 8-dimensional linear mapping between two such light fields. Our key insight is that we can handle the curse of dimensionality inherent in this problem by slightly modifying the long light paths, while keeping the short light paths intact. This way we make the problem tractable while maintaining accuracy.

We use these insights to develop a precomputation-based method for rendering volumetric media with repeated structures. Our algorithm separates low-order and high-order scattering, and modularly precomputes the latter. We model the volumetric medium as a grid of *voxelized blocks*. While a full volume may consist of millions of blocks, there are only a few (tens or hundreds) unique *exemplar* blocks required to synthesize it [112]. We precompute the voxel-to-voxel, patch-to-patch, and patch-to-voxel flux transfer matrices for each of these exemplar blocks. At render time, we modularly stitch together light transport using the precomputed matrices to efficiently compute higher-order scattering. Our contributions include:

- A new formulation for the efficient and tractable rendering of anisotropic volumes by exploiting modularity, and avoiding the curse of dimensionality.
- A Monte Carlo matrix inversion based algorithm to make our approach tractable for very large volumes with millions of blocks, each with thousands of voxels.

- Results demonstrated on a variety of highly complicated volumes, with a focus on cloth (Figures 6.1, 6.14, and 6.15), but generalizing to non-cloth synthesized volumes (Figure 6.16) as well.

We show that this approach can, in many cases, accelerate rendering over path tracing by an order of magnitude. It also significantly outperforms techniques like photon mapping on these media. While the previous research on highly complex measured volumetric materials generated significant interest, it was too expensive in practice. We believe that our method, which runs fast enough on a single server with relatively low memory requirements, will make these materials usable in demanding industry applications including interior design and digital effects.

6.2 Background: path integral formulation

A powerful path formulation of the volume rendering equation (2.2) was introduced by Pauly et al. [78] as an extension of Veach's work [95] on surface-based rendering. The intensity of a pixel in the rendered image is expressed as an integral over all light paths in the scene passing through this pixel.

$$I = \int_{\Omega} f(\bar{x}) \, \mathrm{d}\mu(\bar{x}). \tag{6.1}$$

Here μ is a measure on the path space $\Omega = \bigcup_{l \geq 1} \Omega_l$, and Ω_l is the space of paths with l segments, $\bar{x} = \mathbf{x}_0 \mathbf{x}_1 \ldots \mathbf{x}_l$, such that $\mathbf{x}_0$ is the camera position, $\mathbf{x}_1$ is the surface point or volumetric scattering location directly visible through the pixel, $\mathbf{x}_l$ is on a light source and $\mathbf{x}_2, \ldots, \mathbf{x}_{l-1}$ are any light bounce points in the scene.

The contribution $f(\bar{x})$ of any single path $\bar{x} = \mathbf{x}_0 \mathbf{x}_1 \ldots \mathbf{x}_l$ is the product of a *pixel weight term* $W(\mathbf{x}_0 \leftarrow \mathbf{x}_1)$, a *light emission* term $L_e(\mathbf{x}_{l-1} \leftarrow \mathbf{x}_l)$, *geometry terms*

$G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})$ corresponding to each segment of the path, and *material terms* $M(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k \leftarrow \mathbf{x}_{k+1})$ corresponding to every interior vertex. Geometry terms contain exponential extinction (inside volumes) and $1/r^2$ falloff when necessary. Note also that cosine terms are included in G for surface but not volume events. In the case of volumetric media, the material terms contain a (possibly anisotropic) phase function evaluation:

$$M(\mathbf{x}, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2) = \alpha(\mathbf{x})\, \sigma_t(\mathbf{x}, \boldsymbol{\omega}_1)\, p(\mathbf{x}, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2) \tag{6.2}$$

We assume the albedo $\alpha(\mathbf{x})$ is not directionally dependent. The material term is reciprocal, though the phase function is generally not; as detailed by Jakob et al. [43], the phase function observes the slightly more involved reciprocity relationship $\sigma_t(\mathbf{x}, \boldsymbol{\omega}_1)\, p(\mathbf{x}, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2) = \sigma_t(\mathbf{x}, \boldsymbol{\omega}_2)\, p(\mathbf{x}, \boldsymbol{\omega}_2, \boldsymbol{\omega}_1)$. We use the microflake phase function [43, 111], which satisfies this relationship, in addition to other desirable properties.

We use this path formulation to describe our method. Note that path integrals can also be easily applied to paths that do not have endpoints on the camera or light source.

6.3 Overview

In this section, we describe the main computational difficulty in rendering complex, optically thick media with anisotropic phase functions. We introduce two key simplifications to make the problem tractable: isotropy and diffuser events in long light paths. Based on these simplifications, we introduce a modular approach, which precomputes the light transport within blocks of the volume. The modularity is inspired by Loos et al.'s [62] approach. However, there are differ-

$\bar{x}$	a light path (sequence of vertices)
$\mu(\bar{x})$	measure on path space (product of surface and volume measures on vertices)
$f(\bar{x})$	product of geometry and material terms along path, pixel weight (for camera vertices) and light emission (for light vertices)
$G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)$	geometry term (contains $1/r^2$ falloff when necessary, volume extinction, and cosine terms for surface, but not volume vertices)
$M(\mathbf{x}, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2)$	material term: a product of the phase function $p(\mathbf{x}, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2)$, the single-scattering albedo $\alpha(\mathbf{x})$, and the extinction coefficient $\sigma_t(\mathbf{x}, \boldsymbol{\omega}_1)$
B_i	block: a grid of voxels
V_i	voxel: at precomputation level, not data level
N_i	non-empty subset of voxel V_i
$\lvert N_i \rvert$	volume of N_i
P_i	patch: an oriented voxel face on the shared interface between two blocks
$\mathrm{flip}(i)$	flip operator on set of patch indices
Q	a permutation matrix encoding $\mathrm{flip}(i)$
T_i^{vv}, T_i^{vp}, T_i^{pv}, T_i^{pp}	voxel-to-voxel, voxel-to-patch, patch-to-voxel, and patch-to-patch transfer matrices for block B_i
$\tilde{T}^{vv}$, $\tilde{T}^{vp}$, $\tilde{T}^{pv}$, $\tilde{T}^{pp}$	global completions of transfer matrices
$\boldsymbol{\Phi}^s$	source flux: integral of illumination up to first scattering event
$\boldsymbol{\Phi}^{gt}$	ground truth multiple-scattered flux
$\boldsymbol{\Phi}^m$	multiple-scattered flux (approximation of $\boldsymbol{\Phi}^{gt}$ computed by modular transfer)

Table 6.1: Summary of notation; bold letters indicate vectors.

ences in the main challenges that need to be handled. While Loos et al.'s main challenge is compression of the transport into very small vectors of coefficients that allow for real-time evaluation, we instead need to deal with extremely large resolution and long light paths.

Key challenge: long paths. The volumetric path integral can be naturally

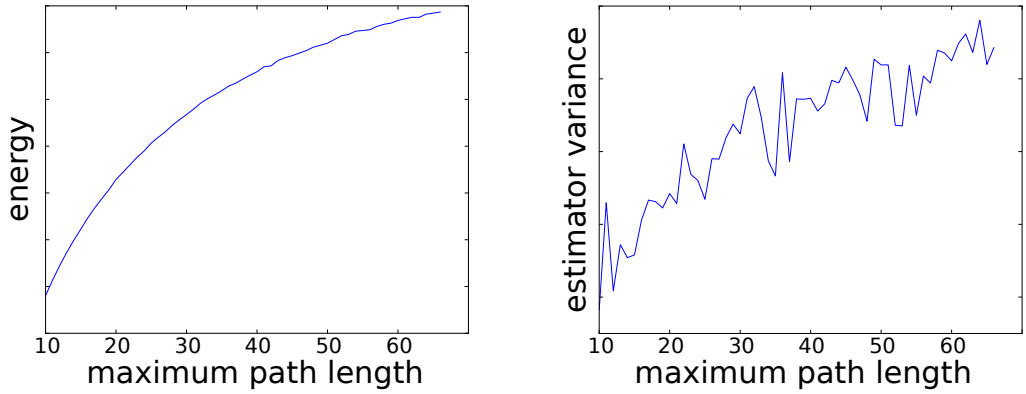
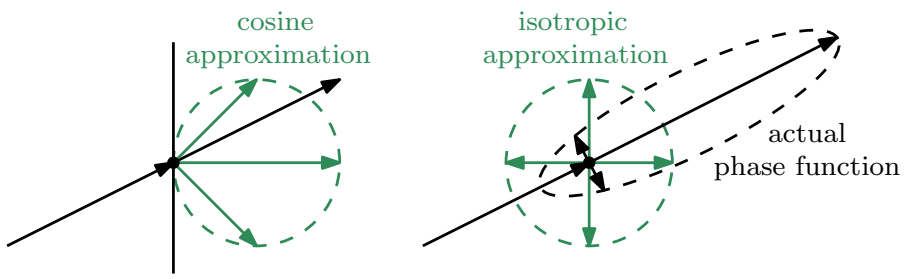


Figure 6.2: The increase in the total energy of an image, and the corresponding increase in the variance of a Monte Carlo path tracer, as a function of the maximum number of scattering events (with number of samples held constant). The data is measured on our felt scene (top of Figure 6.14), where the green single-scattering albedo has been set to 0.99.

evaluated by Monte Carlo sampling of paths with known probabilities, which leads to a standard volumetric path tracer. The problem with this approach is its slow performance, especially for materials with highly complicated structures, such as data-driven cloth, where creating each path vertex executes a Woodcock tracking procedure to importance-sample extinction, and has a significant cost of many non-cached memory accesses. Furthermore, a significant amount of energy (and resulting variance) is in paths with many segments. This problem is most significant for bright colors, which have a high single-scattering albedo in one or more color channels. Figure 6.2 shows the increase in energy in a cloth sample if albedo is high, and the corresponding increase in variance in a path tracing solution.

Diffuser and isotropy events. Shorter paths can be handled by pure path tracing; like some previous approaches, we apply our approximations to longer paths. The key difference is that we take advantage of the repeated structure in the volume to *precompute* a large number of paths, driving the effective num-



(a) diffuser event (b) isotropy event

Figure 6.3: Inserting **isotropy events** and **diffuser events** into paths makes the underlying path integral separable, at the cost of introducing some error. If these events are inserted into paths of sufficient length, the error will be close to imperceptible.

ber of paths significantly higher (and the noise much lower) than any non-precomputed approach.

One key challenge is to split paths into precomputed and runtime components without introducing high-dimensional (and therefore expensive) intermediate data representations. The coupled nature of the material terms along a path through an anisotropic volume makes any attempt to split the problem into subproblems (some of which could be precomputed) seem hopeless.

We propose modifying the problem by inserting *diffuser events* and *isotropy events* into some paths (see Figure 6.3). We found that setting the phase function on a small number of vertices of a long path to be isotropic creates very little error in the final image, which makes precomputation feasible. Therefore, we insert isotropy events on the k-th vertex from the camera and the last vertex before the light. Also, we insert zero or more diffuser events in between. These modifications provide significant advantages by making the underlying path integrals separable, in the sense that they can be factored into a product of integrals "before" and "after" the isotropy/diffuser event. This allows us to split the paths into components that are precomputed offline, and computed

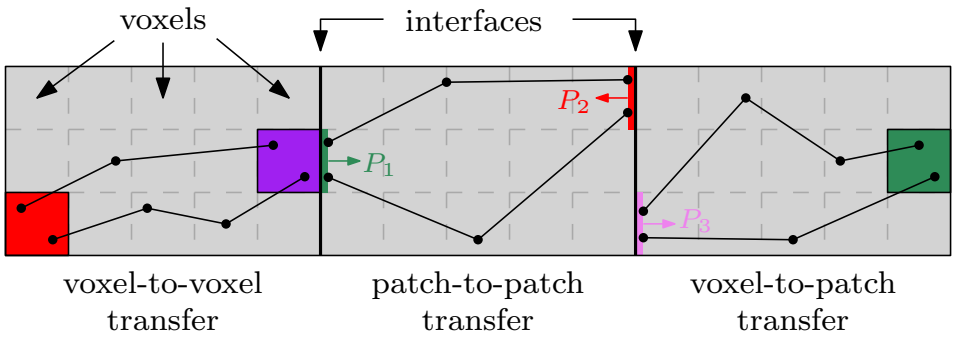


Figure 6.4: Definitions of **voxels**, **interfaces**, **patches**, and three types of **precomputed transfers**, each of which corresponds to a matrix. Note that patch-to-voxel transport is the transpose of voxel-to-patch.

at runtime. In other words, this enables us to use *flux* instead of *light fields* as the quantity being transported, addressing the curse of dimensionality, and making the problem tractable. These assumptions are later used in Sections 6.5.1 and 6.5.3. In comparison, modular radiance transfer [62] uses low-resolution lightfields at block boundaries. We considered coarse directional discretizations, but found their storage to be too expensive: for example, 32 directional bins would inflate our precomputed data 1,024 times.

We show that, if done for paths of sufficient length, these modifications have little effect on the accuracy of the solution. In fact, as our results show, the accuracy of our results is higher with a practical number of samples than with methods like path tracing or photon mapping. While theoretically these approaches do not make any isotropy or diffuser approximations, in practice they need many more samples to achieve a quality as high as our results (Section 6.6.2).

6.3.1 Definitions

Blocks. We divide the volume into a number of equal-sized *blocks*. Our algorithm makes no assumption about the definition of a block. For our woven fabric

data, a block is defined to be of size about 0.5 mm (approximately a yarn crossing, as suggested in [112]). We treat the volume as consisting of a 2-dimensional grid of blocks without loss of generality; the grid could also easily be 3-dimensional, but in our results there is always only one block across the thickness of the volume). The full volume can be made of millions of blocks, but due to repeating structures, only a small number (up to one or two hundreds in our case) of the blocks are unique, and we call them *exemplar blocks*.

Voxels. Each block is divided into n *voxels* $V_1, V_2, \ldots, V_n$, which provide the resolution at which precomputed transfer occurs. Note that these voxels are separate from the underlying volume's representation, which can have much higher resolution; i.e., within each precomputed transfer voxel, there can still be many underlying data voxels. We define the *non-empty subset* of voxel V_i as $N_i := \{\mathbf{x} \in \tilde{V}_i \mid \sigma_t(\mathbf{x}, \boldsymbol{\omega}) > 0 \text{ for some } \boldsymbol{\omega}\}$ where $\tilde{V}_i \subset \mathbb{R}^3$ denotes the 3D space contained in voxel V_i.

Interface. We define an *interface* to be the rectangular shared boundary between two neighboring blocks. The final volume can contain millions of distinct interfaces.

Patches. We define *patches* $P_1, P_2, \ldots$ to be oriented faces of voxels on an interface. Oriented-ness means that each patch can be associated with a normal vector $\mathbf{n}(P_i)$, pointing into one of the neighboring blocks of the underlying interface. Patches will serve the purpose of connecting the transfer between blocks.

Figure 6.4 shows a flatland visualization of a volume consisting of three blocks, each of which is divided into 3×5 voxels. In this example, there are two interfaces and twelve patches. This corresponds directly to our actual algorithm,

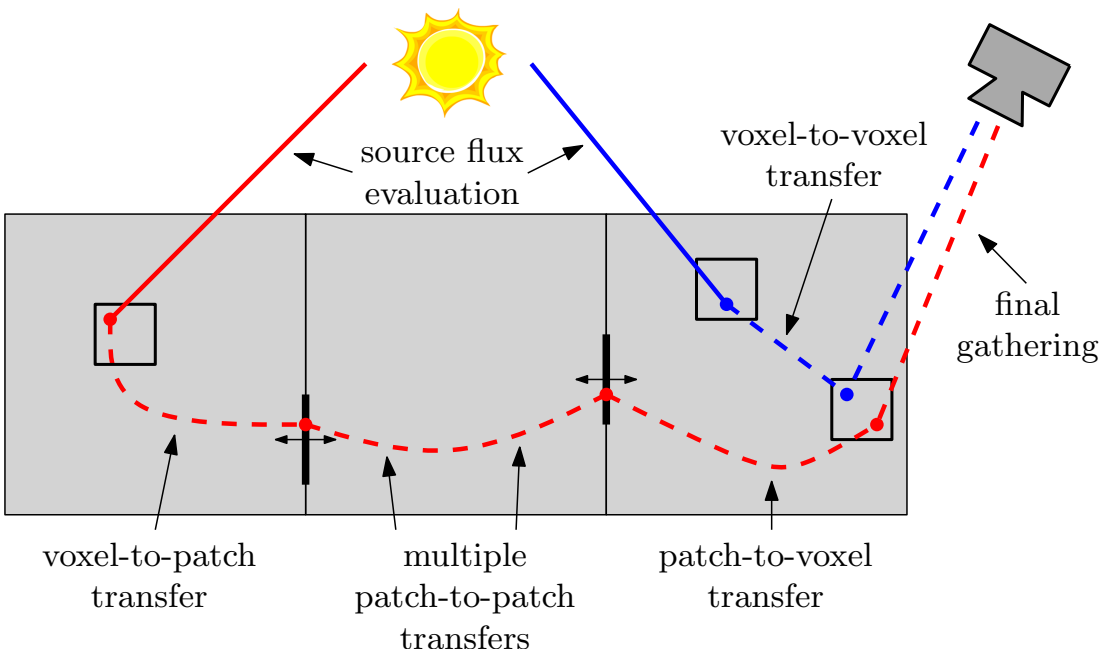


Figure 6.5: **The three phases of the runtime stage of our pipeline.** We use dashed lines to indicate sub-paths with length ≥ 1 which can contain multiple scattering events.

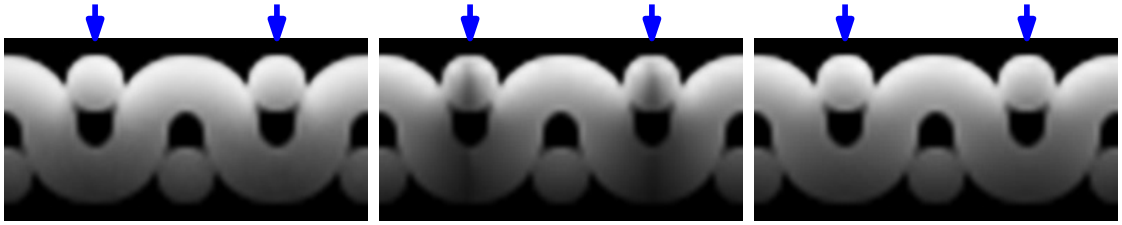


Figure 6.6: **Cropped 2D slices of the flux field of a synthetic volume** with three blocks where the interfaces are indicated with blue arrows: (left) path-traced reference; (center) applying precomputed voxel-to-voxel transfer within blocks leads to darkening, because paths crossing the boundaries are missing; (right) adding transfer across boundaries addresses this energy loss.

except the real implementation is in three dimensions, uses millions of blocks, and is optionally warped by a shell-map [83] (to bend the volume into a curved shape).

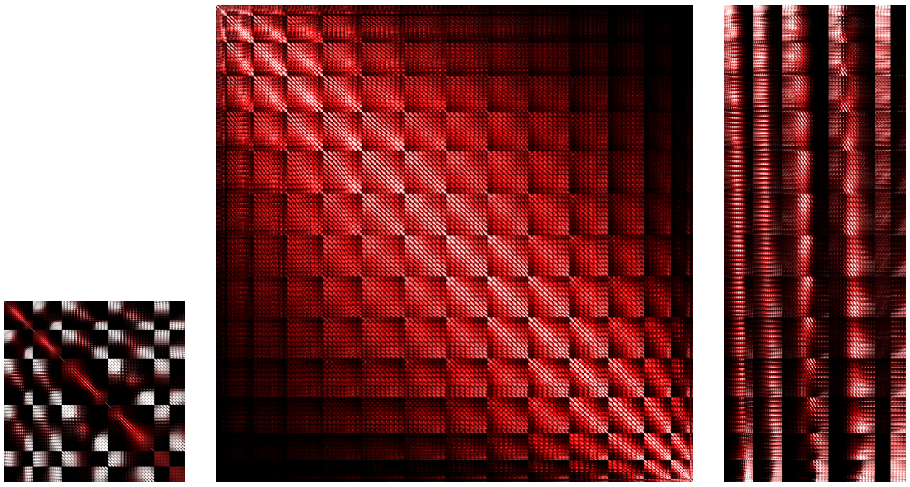
6.3.2 Modular transfer pipeline

The pipeline of our system consists of the following stages:

Transfer matrix precomputation. In this stage, we pre-compute the light transfer for a set of exemplar blocks which can later be used to assemble full high-resolution volumes using various techniques including [112]. In particular, we compute three types of transfers, voxel-to-voxel, voxel-to-patch, and patch-to-patch, which will be introduced in Section 6.4.

Runtime evaluation. At the runtime stage, the pipeline is split into the following three phases (see Figure 6.5):

1. **Source flux evaluation.** We first evaluate the amount of attenuated light arriving at voxels directly from a light source. This can be treated as the first scattering event, and will serve as the source term for the light propagation of precomputed transfer matrices. Note that, conceptually, the source flux includes the scattering event (more precisely, the material term).
2. **Modular transfer.** Given the source flux, we apply our precomputed transfer to obtain multiple-scattered flux. The voxel-to-voxel transfer captures light transfer within individual blocks, and accounts for the bulk of short range transport. Further, voxel-to-patch and patch-to-patch transfers provide longer-range transport that crosses block boundaries. All of these transfers are introduced in Section 6.5.2.
3. **Final gathering.** Finally, we run a standard path tracing algorithm accelerated by looking up the previously computed scattered flux field after k scattering events (paths with less than k scatterings are computed by explicit path tracing). More details are available in Section 6.5.3.



(a) (b) (c)

Figure 6.7: **Visualizations of precomputed transfer matrices** of a twill block with 470 patches and 1239 non-empty voxels: (a) patch-to-patch, (b) voxel-to-voxel, and (c) patch-to-voxel.

6.4 Precomputation

In this section, we describe the precomputation stage of our pipeline. Given a block divided into n voxels, we define three kinds of transfer matrices: voxel-to-voxel, voxel-to-patch, and patch-to-patch. Below we give precise definitions of the elements of the transfer matrices, and describe a Monte Carlo particle tracing algorithm to compute them.

Voxel-to-voxel. The *voxel-to-voxel* transfer matrix T^{vv} handles most (but not all) of the light transport: because of the short mean free path in optically dense materials, most transport is local, even though it requires many scattering events. Figure 6.6 compares the amount of transport contained within the voxel-to-voxel transfer versus the other transfer modes.

The element (i, j) of T^{vv} is defined as a path integral, where the endpoints of

the paths are in N_i and N_j, the non-empty subsets of V_i and V_j:

$$T^{vv}(i,j) = \int_{\Omega(N_i,N_j)} f(\bar{x})\,\mathrm{d}\mu(\bar{x}). \tag{6.3}$$

Here $\Omega(N_i, N_j)$ means the set of paths (with one or more segments) with endpoints in N_i and N_j, respectively (see the left block in Figure 6.4). This definition implies that T^{vv} is symmetric. The use of the non-empty subsets N_i instead of the full voxels V_i is important for several reasons. It leads to sparser matrices (and thus less storage). Furthermore, defining the path integral this way helps us to compute total flux within non-empty regions of a voxel, which approximates the radiance values at scattering events better (since these never occur in empty regions).

We compute T^{vv} using Monte Carlo particle tracing, as shown in Algorithm 6.1. For each voxel V_i, we trace m paths. For each of the m paths we trace the path through the block, with appropriate importance sampling, and deposit values into the appropriate (i, j) entry in the transport matrix T_{vv}. This is done inside the loop on lines 6-18. Each path terminates when it exits the volume (line 12). At each vertex, energy is deposited (line 14) and then multiplied by the albedo (line 15). Note that the deposition is divided by the probability density of the initial particle generation, captured in the initial weight $w = |N_i|$ for position choice (where $|N_i|$ is the volume of the non-empty voxel subset), and 4π for direction choice. The value is also divided by the value of σ_t at the deposition vertex, because it figures in the sampling probability density but not in the path integral we want to compute. Since $\mathbf{x}'$ is chosen by importance-sampling σ_t, it holds that $\sigma_t(\mathbf{x}', \boldsymbol{\omega}) > 0$ in line 14, preventing any division-by-zero issues.

In summary, the particle tracing importance-samples all terms along a path other than the initial probability, albedo terms, and the final division by σ_t, which

Algorithm 6.1 A particle tracing based method for computing the voxel-to-voxel transfer matrix T^{vv} given a block B.

$T^{vv} \leftarrow \mathbf{0}$
for $i = 1$ to numVoxels **do** ▷ Loop over all voxels
  for $t = 1$ to m **do**
    sample a ray $(\mathbf{x}, \boldsymbol{\omega})$ with $\mathbf{x} \in N_i$ and $\boldsymbol{\omega} \in \mathcal{S}^2$
    $w \leftarrow 4\pi|N_i|$ ▷ Weight initialization
    loop
      sample s (with Woodcock tracking) according to

$$p(s) = \sigma_t(\mathbf{x} + s\boldsymbol{\omega}, \boldsymbol{\omega}) \exp\left(-\int_0^s \sigma_t(\mathbf{x} + s'\boldsymbol{\omega}, \boldsymbol{\omega})\, \mathrm{d}s'\right)$$

      $\mathbf{x}' \leftarrow \mathbf{x} + s\boldsymbol{\omega}$ ▷ Get a scattering event at $\mathbf{x}'$
      if $\mathbf{x}' \in B$ **then**
        find the voxel j that contains $\mathbf{x}'$
      else
        break ▷ Terminate the path
      end if
      $T^{vv}(i, j) \leftarrow T^{vv}(i, j) + w/\sigma_t(\mathbf{x}', \boldsymbol{\omega})$ ▷ Deposit energy
      $w \leftarrow w \cdot \alpha(\mathbf{x}')$ ▷ Update the weight
      sample direction $\boldsymbol{\omega}'$ according to material term at $\mathbf{x}'$
      $(\mathbf{x}, \boldsymbol{\omega}) \leftarrow (\mathbf{x}', \boldsymbol{\omega}')$ ▷ Continue the path
    end loop
  end for
end for
$T^{vv} \leftarrow T^{vv}/m$

are exactly the terms that occur explicitly in the weighting.

Voxel-to-patch. As shown in the right block of Figure 6.4, the *voxel-to-patch* transfer matrix T^{vp} is similarly defined as the integral over paths between a voxel V_i and patch P_j:

$$T^{vp}(i, j) = \int_{\Omega(N_i, P_j)} f(\bar{x})\, \mathrm{d}\mu(\bar{x}), \tag{6.4}$$

where $\Omega(N_i, P_j)$ is again naturally defined as the set of paths with endpoints in N_i and P_j. Importantly, the non-empty voxel subset N_i is used again. Note the

contribution $f(\bar{x})$ includes a cosine term on the path vertex that lies on the patch. This is because geometry terms include a cosine on surfaces, and the patch is treated as a surface. Note that patch-to-voxel transport T^{pv} is the transpose of T^{vp}, so no separate definition is required for it. These matrices are normally not square.

The computation of T^{pv} is handled similarly to T^{vv} by volumetric particle tracing, except the origin of a particle is chosen on a patch, and its direction is chosen by importance-sampling the cosine of the angle from the patch normal.

Patch-to-patch. The *patch-to-patch* transfer matrix T^{pp} is defined as the integral over paths between patches P_i and P_j (see the center block in Figure 6.4):

$$T^{pp}(i,j) = \int_{\Omega(P_i,P_j)} f(\bar{x})\,\mathrm{d}\mu(\bar{x}). \tag{6.5}$$

Computing T^{pp} is analogous to T^{vp}, except deposition occurs at other patches instead of voxels. Matrix T^{pp} is also symmetric.

Summary of precomputation. Assume we have a set of exemplar blocks which can be assembled to construct very large volumes. For the i-th exemplar block, we precompute and store the voxel-to-voxel, voxel-to-patch, and patch-to-patch matrices T_i^{vv}, T_i^{vp}, and T_i^{pp}. Figure 6.7 visualizes the transfer matrices of a block of twill fabric with shiny red fibers.

6.5 Runtime evaluation

In this section, we describe the three phases of the runtime stage of our pipeline: source flux evaluation, modular transfer, and final gathering. The first and third

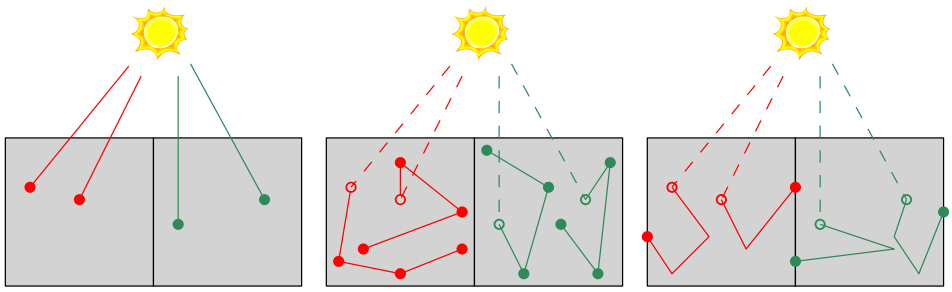


Figure 6.8: Light paths captured by (a) source flux $\mathbf{\Phi}^s$; (b) $\mathbf{\Phi}^s$ with voxel-to-voxel transfer applied; (c) $\mathbf{\Phi}^s$ with voxel-to-patch transfer applied.

stages are based on standard approaches, while the core complexity lies in the second phase, where the precomputed transfers are applied.

6.5.1 Source flux evaluation

The runtime of our pipeline starts by evaluating the *source flux vector* $\mathbf{\Phi}^s$, which is the amount of single-scattered light arriving at voxels directly from the set of direct light sources (see Figure 6.8a). Denote the set of emissive surfaces C. Let $\Omega_1(N_i, C)$ be the set of unscattered paths (i.e., direct single-segment connections) from voxel subsets N_i to the set of emissive surfaces C. Denote these single-segment paths by $\bar{x} = \mathbf{x}_0\mathbf{x}_1$. The elements of $\mathbf{\Phi}^s$ can now be defined as:

$$\Phi^s(i) = \int_{\Omega_1(N_i,C)} \int_{S^2} M(\mathbf{x}_0, \boldsymbol{\omega}, \overrightarrow{\mathbf{x}_0\mathbf{x}_1}) f(\bar{x}) \, \mathrm{d}\boldsymbol{\omega} \, \mathrm{d}\mu(\bar{x}). \tag{6.6}$$

An important detail is that, intuitively, we are including in $\mathbf{\Phi}^s$ light that scattered once. More precisely, we include the material term in the definition, integrating it over all directions. This is the first application of an isotropy event: since we do not know $\boldsymbol{\omega}$, the direction light will take after this scattering event, we integrate over all such directions.

We compute $\mathbf{\Phi}^s$ by particle tracing, and store it as a sparse vector where the

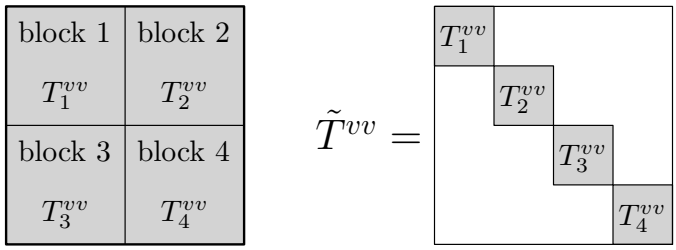


Figure 6.9: Formation of block-diagonal matrix $\tilde{T}^{vv}$ for a volume with four blocks.

total number of non-zero entries never exceeds the number of traced particles, and is usually much smaller than the number of voxels. The noise in $\mathbf{\Phi}^s$ is not a problem, because the application of the precomputed transfer matrices multiplies the effective number of paths by a large number.

6.5.2 Modular transfer

The goal of the modular transfer phase of the pipeline is to use the precomputed matrices to turn source flux into multiple-scattered flux, which can then be simply looked up by the final gather phase of the pipeline. We define the *ground-truth multiple-scattered flux* $\mathbf{\Phi}^{gt}$ as:

$$\mathbf{\Phi}^{gt}(i) = \int_{\Omega_{2+}(N_i,C)} f(\bar{x})\,\mathrm{d}\mu(\bar{x}). \tag{6.7}$$

Here, by $\int_{\Omega_{2+}(N_i,C)}$ we mean paths with 2 or more segments. Our goal is to compute an approximate multiple-scattered flux $\mathbf{\Phi}^m$ by application of precomputed transfer, such that $\mathbf{\Phi}^m \approx \mathbf{\Phi}^{gt}$. An important detail is that $\mathbf{\Phi}^m$, unlike $\mathbf{\Phi}^s$, does not conceptually contain the scattering event, i.e. the material term; it will be the responsibility of the final gather phase to include it.

For the final volume (where each block is the copy of an exemplar), let the transfer matrices of the i-th block be T_i^{vv}, T_i^{vp} and T_i^{pp}. We define the *global*

completions of these matrices, which operate on a global indexing of all voxels and patches in the volume, and denote them by $\tilde{T}^{vv}$, $\tilde{T}^{vp}$, and $\tilde{T}^{pp}$. If we assign contiguous indices to voxels belonging to the same block and patches belonging to the same interface, then $\tilde{T}^{vv}$ and $\tilde{T}^{pp}$ become block diagonal (see Figure 6.9). In other words, this is simply an imaginary stacking of the precomputed matrices for different exemplar blocks, so that we have only one large block-diagonal matrix for each transfer type.

Given the source flux vector $\mathbf{\Phi}^s$, the first step of the modular transfer phase is the application of voxel-to-voxel transfer. This is accomplished simply by computing the matrix-vector multiplication $\tilde{T}^{vv}\mathbf{\Phi}^s$ (see Figure 6.8b).

However, the voxel-to-voxel matrix $\tilde{T}^{vv}$ only encodes transfer over paths that do not cross block interfaces, as shown in Figure 6.8b. To compensate for this omission, we need to propagate the flux to the block interface, "cross" to the other side of the interface to enter the neighboring block, and then propagate further to voxels in that block.

More precisely, we transfer voxel fluxes into patch fluxes by an application of $\tilde{T}^{vp}$ (see Figure 6.8c). Now the patch flux on patch P_i describes the amount of light it receives. Let $\mathrm{flip}(i)$ denote the index of the patch that overlaps with i but with the opposite normal direction (so patches P_i and $P_{\mathrm{flip}(i)}$ always belong to immediately neighboring blocks). This flip operator can be written as a permutation matrix Q on the set of global patch indices. To propagate the patch flux received by P_i, we make the opposite patch $P_{\mathrm{flip}(i)}$ to emit the same amount of energy into the neighboring block (see Figure 6.10ab): this is an application of a diffuser event, which prevents the need to store a directionally-dependent function on each patch; instead it suffices to compute a scalar quantity, the patch flux. Note that this assumption is crucial to making our approach tractable since

storing directional variation would prohibitively increase the amount of storage. However, this still does not account for all light paths. To allow the energy to go further, we need to apply $Q\tilde{T}^{pp}$ multiple times, to go from patch to patch across entire blocks (see Figure 6.10cd). This will add the light traversing block boundaries for longer range transport. This eventually leads to the correct expression for computing the multiple-scattered flux:

$$\mathbf{\Phi}^m = \tilde{T}^{vv}\mathbf{\Phi}^s + \tilde{T}^{pv}\left(\sum_{a=0}^{\infty}\left(Q\tilde{T}^{pp}\right)^a\right)Q\tilde{T}^{vp}\mathbf{\Phi}^s. \tag{6.8}$$

Let $U := I - Q\tilde{T}^{pp}$. Rewriting the infinite series in (6.8) using the Neumann series yields:

$$\mathbf{\Phi}^m = \tilde{T}^{vv}\mathbf{\Phi}^s + \tilde{T}^{pv}U^{-1}Q\tilde{T}^{vp}\mathbf{\Phi}^s, \tag{6.9}$$

which can be computed as follows. First, we compute by solving a linear system the *patch flux*:

$$\mathbf{\Phi}^p := U^{-1}Q\tilde{T}^{vp}\mathbf{\Phi}^s, \tag{6.10}$$

which equals the amount of energy emitted by each patch i contributed by light paths going across one or more interfaces and flipping across the interface. Then we evaluate (6.9) using $\mathbf{\Phi}^m = \tilde{T}^{vv}\mathbf{\Phi}^s + \tilde{T}^{pv}\mathbf{\Phi}^p$.

To compute Equation (6.10), we need to solve a large linear system $U\mathbf{\Phi}^p = Q\tilde{T}^{vp}\mathbf{\Phi}^s$. Since U is usually asymmetric (because of the flip operator Q), conjugate gradient based algorithms cannot be applied here. Instead, we introduce two methods corresponding to finite element and Monte Carlo approaches to solve this system.

Jacobi Iteration. We can simply truncate the Neumann series, which results in Jacobi iteration. We start with $\mathbf{\Psi}^{(0)} = Q\tilde{T}^{vp}\mathbf{\Phi}^s$, and in the t-th iteration, we compute $\mathbf{\Psi}^{(t+1)} = Q\tilde{T}^{pp}\mathbf{\Psi}^{(t)} + \mathbf{\Psi}^{(0)}$. After a few iterations, set $\mathbf{\Phi}^p = \mathbf{\Psi}^{(t)}$. Note

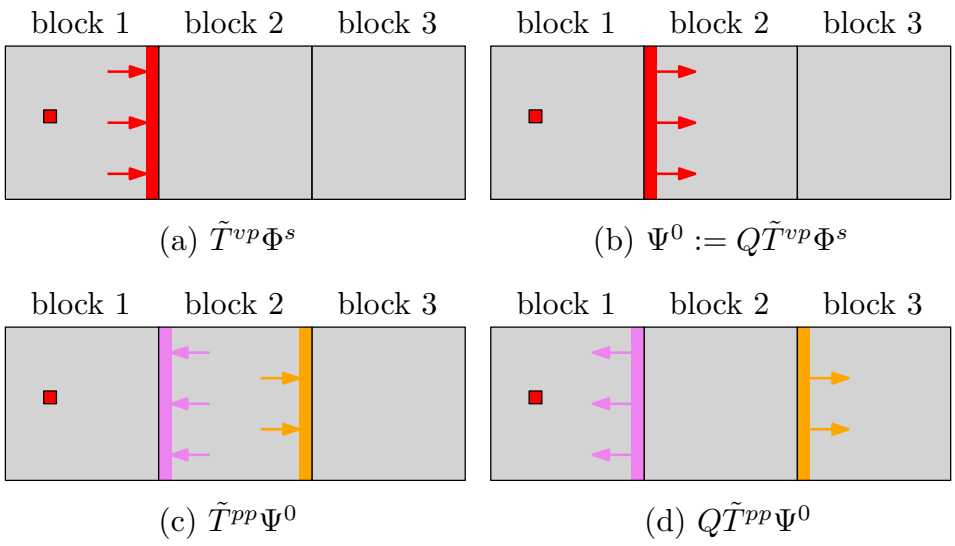


Figure 6.10: **An example of patch flux propagation.** The scene contains 3 blocks and 4 patches defined over 2 interfaces. Assume that all voxels have zero flux except for one in block 1 marked with the red square. Then (a) shows the patch flux received by the right patch in block 1; (b) applying flip operator Q gives patch flux emitted by the left patch in block 2; (c) multiplying by $\tilde{T}^{pp}$ gives the patch flux received by both patches in block 2; (d) applying another Q yields the patch flux emitted by the two patches in blocks 1 and 3.

that neither $\tilde{T}^{pp}$ nor Q needs to be formed explicitly; instead, for any given vector $\mathbf{v}$, $\tilde{T}^{pp}\mathbf{v}$ can be computed by performing the patch-to-patch transfer on each block, and $Q\mathbf{v}$ can be obtained by permuting $\mathbf{v}$'s elements.

Monte Carlo Matrix Inversion. Although Jacobi iteration works adequately, it may take many iterations to converge and requires storing the full vector $\mathbf{\Psi}^{(t)}$ (whose size equals the total number of patches), which can be very expensive: the tablecloth in Figure 6.1, for example, contains many millions of block interfaces, each with hundreds of patches. To make our method truly scalable to very large models, we implemented a Monte Carlo method based on [24], which does not require the storage of $\mathbf{\Phi}^p$ explicitly. Let

$$\mathbf{\Phi}^{ls} := \tilde{T}^{pv}\mathbf{\Phi}^p = \tilde{T}^{pv}U^{-1}Q\tilde{T}^{vp}\mathbf{\Phi}^s, \tag{6.11}$$

Algorithm 6.2 Random walk estimating $\mathbf{\Phi}^{ls}(i)$.

$(j, s) \leftarrow \text{sampleRow}(\tilde{T}^{pv}, i)$ ▷ Jump from voxel to patch
loop
  $j \leftarrow \text{flip}(j)$ ▷ Flip the patch
  Let q be the stop probability
  if $\text{rand}() < q$ **then** ▷ Russian roulette
    $s \leftarrow s/(1-q)$
    break
  end if
  $(j, v) \leftarrow \text{sampleRow}(\tilde{T}^{pp}, j)$ ▷ Jump to another patch
  $s \leftarrow sv/q$ ▷ Update the throughput
end loop
$(i', v) \leftarrow \text{sampleRow}(\tilde{T}^{vp}, j)$ ▷ Final jump to voxel
return $\mathbf{\Phi}^s(i')\, sv$

which is the component of the multiple-scattered flux that requires solving a linear system (i.e., summing a Neumann series). If elements of $\mathbf{\Phi}^{ls}$ can be efficiently estimated, we can then compute Equation (6.9) with $\mathbf{\Phi}^m = \tilde{T}^{vv}\mathbf{\Phi}^s + \mathbf{\Phi}^{ls}$.

To do this, we define a *random walk* whose expected value is the i-th element of $\mathbf{\Phi}^{ls}$. The random walk intuitively traces a "path", whose "vertices" are discrete states corresponding to voxel and patch indices, instead of actual scattering points, and its "edges" consist of multi-vertex jumps precomputed within the transfer matrix elements.

The algorithm conceptually starts at voxel V_i, and immediately transitions into the j_1-th patch, with a discrete probability distribution proportional to the i-th row of $\tilde{T}^{pv}$. Then, a series of transitions between patches with indices j_1, j_2, j_3, ... is made with probabilities proportional to a corresponding element of the matrix $\tilde{T}^{pp}$. A Russian roulette approach is used to eventually terminate this loop, making a final transition to a voxel index i', with probability based on the corresponding row of $\tilde{T}^{vp}$. At this point, the i'-th element of source flux $\mathbf{\Phi}^s$ is queried and scaled by terms accumulated along the path. It is not difficult

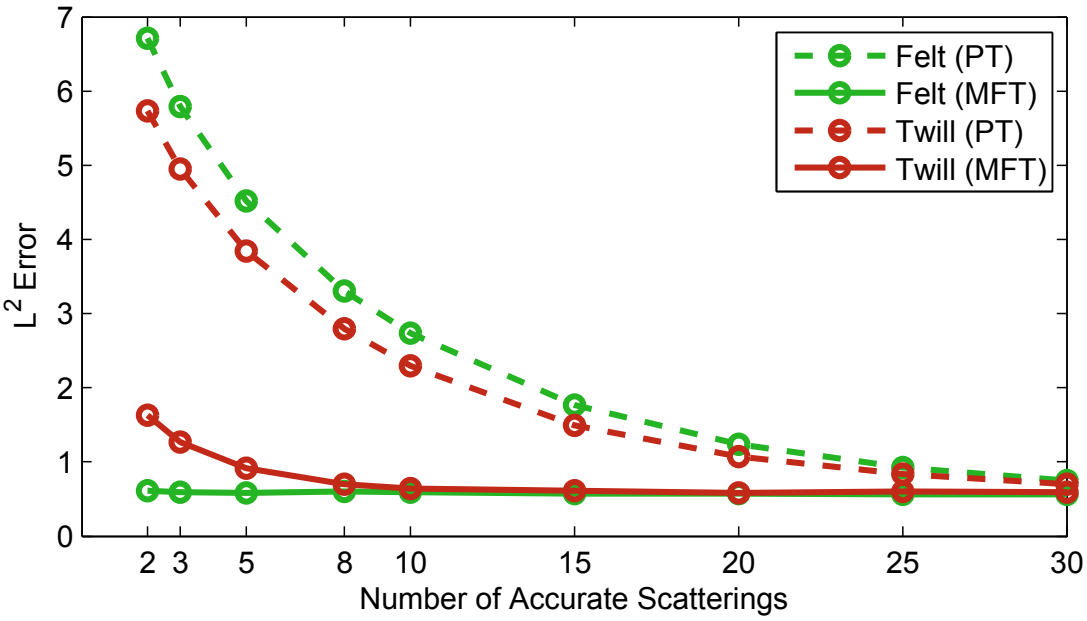


Figure 6.11: **Convergence experiment:** we rendered multiple results with our method (solid lines) and path tracing with terminating the path after k scatterings (dashed lines) using varying k values and computed their $\mathcal{L}^2$ error (plotted as $\log(1+y)$ for error y). Note that the graphs do not converge to zero, because there is Monte Carlo noise in both images being compared.

to see that the expected value of this process is precisely the matrix expression containing the Neumann series that we are interested in computing. Note that each transition in this random walk can capture many light scattering events which are expensive to simulate exactly.

Algorithm 6.2 describes the random walk in more detail. In the algorithm, we use a sampleRow(A, i) function, which returns the index of an element sampled from the i-th row of matrix A, with a probability proportional to the element's value. It also returns the value of the element, divided by the probability of choosing it. All the rendered results in Section 6.6 are generated using this algorithm.

6.5.3 Final gathering

Given the discretized flux field inside the volume computed by the modular transfer, we can now render it efficiently. We perform a standard Monte Carlo path tracing algorithm with explicit direct illumination. However, when handling the k-th scattering event on a light path at location $\mathbf{x}$, we replace the phase function by a uniform isotropic one. This is the second time we insert an isotropy event to obtain a separable approximation to the path integral. This lets us stop the recursive process and approximate the indirect illumination at point $\mathbf{x}$ as $\mathbf{\Phi}^m(i)/(4\pi|N_i|)$ where $\mathbf{\Phi}^m(i)$ is the stored multiple-scattered flux value computed by the modular transfer. The final gather computes all paths with less than k scattering events using standard path tracing.

Finally, we choose the value of k as follows. As shown in Figure 6.11, for a range of materials, we produced multiple renderings with changing k values. The results indicate that for materials with random structures, such as felt, a k value of 2 or 3 is sufficient. For structured materials with parallel shiny fibers (such as the twill), a value of $k = 6$ produces high quality results, and our approach still converges much faster than path tracing. Therefore, we picked $k = 6$ for all our results.

6.6 Results

In this section, we first demonstrate the performance of our technique by comparing flux fields computed by our method, path tracing, and photon mapping. Afterwards we show renderings for a range of materials created with our method.

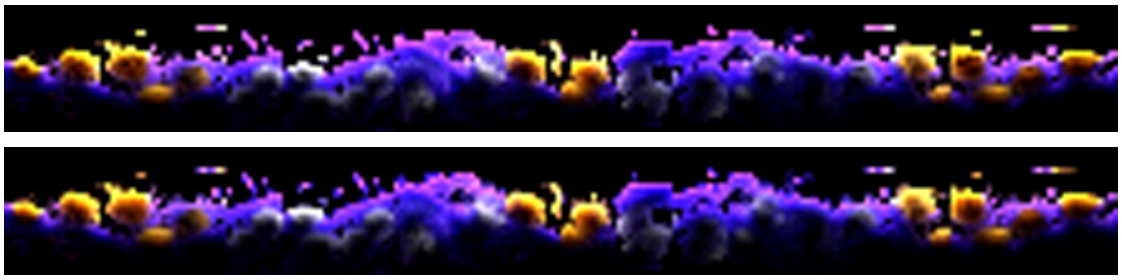


Figure 6.12: **2D slices of flux fields** in non-empty voxels computed with path-tracing (top) and MFT (bottom).

6.6.1 Flux field visualizations

Figure 6.12 shows the multiple-scattered flux fields obtained by path tracing and our technique. The entire volume contains over one million blocks, and in this figure we show 2D slices across 20 of them. The top of the figure shows the reference Φ_{gt} generated by tracing thousands of paths for each non-empty voxel, while the bottom shows the approximate Φ_m generated by tracing 50 million particles for the source flux and solving the modular transfer described in Section 6.5.2. Our result matches the ground truth well.

6.6.2 Photon mapping comparisons

Figure 6.13 shows detail renderings generated by path tracing (the ground truth), our technique (MFT), and photon mapping. Here we use $k = 3$, i.e. three path-traced events precede the MFT or photon map lookup. The entire scene is a single sheet of cloth with over one million blocks, of which a small patch (containing 0.24% of all traced particles) is shown. Figure 6.13b is obtained by tracing 10 million particles for the source flux. Our method is much faster than path tracing while providing good accuracy. Figure 6.13c is generated with photon mapping with 100 million particles. Artifacts result from the stored photon den-

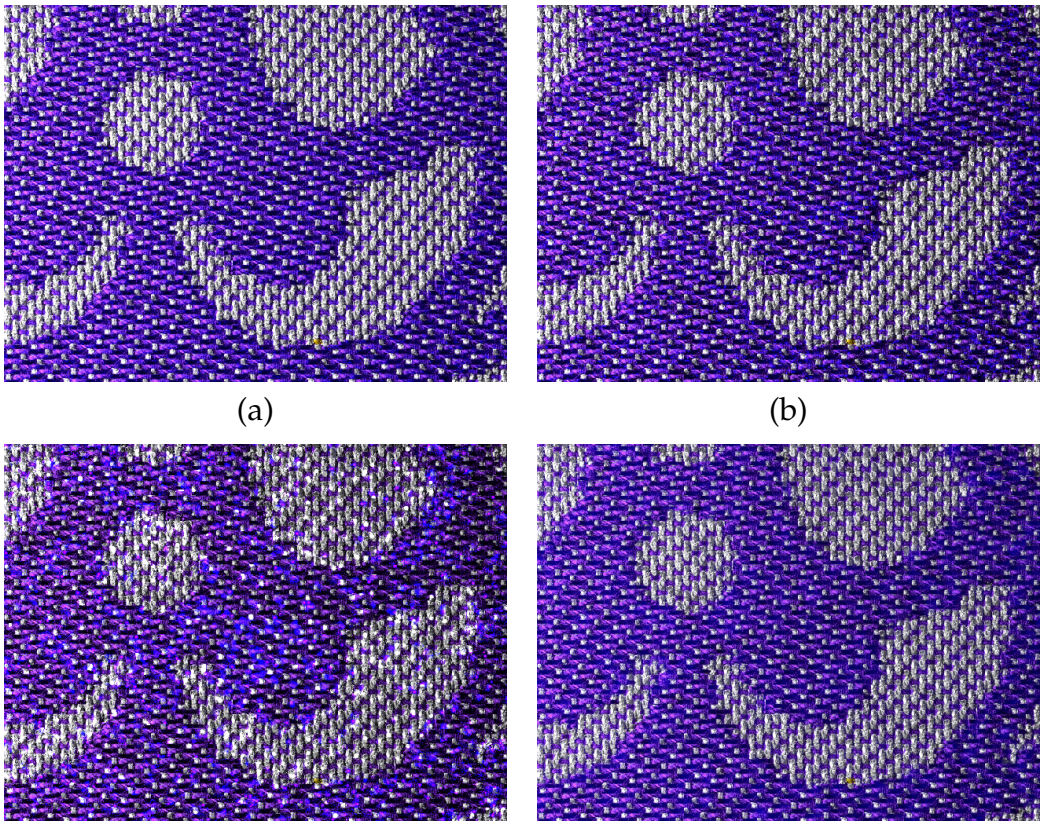


Figure 6.13: Images rendered with (a) standard path tracing in 1.3 hours; (b) MFT in 12 minutes (with 10M particles traced); (c) volume photon mapping in 20 minutes (with 100M photons stored); (d) volume photon mapping in 1 hour (with a billion photons stored).

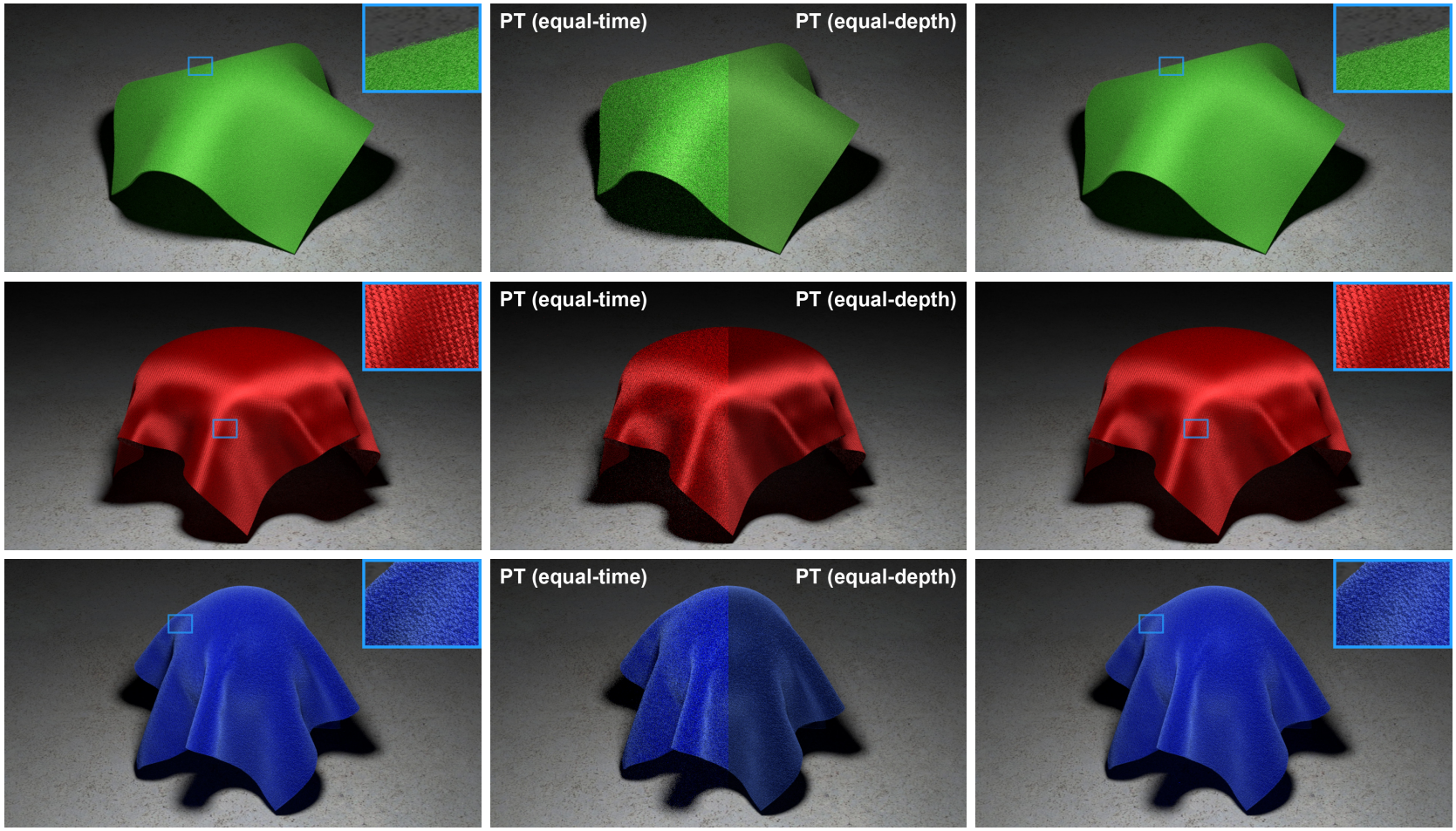
sity being insufficient to capture the yarn-level structures. In Figure 6.13d, we show a photon mapping result using one billion photons.

6.6.3 Rendered results

Next, we show renderings of a variety of volumetric appearance models of fabrics created with techniques introduced in Chapters 4 and 5. These models often consist of trillions of micron-resolution (effective) voxels, making them very challenging to render. In our experiments, we focus on demonstrating

our technique over such highly complex volumetric models. We used a simple lighting configuration and compared our results with those created by volume path tracing. We implemented our system based on the Mitsuba physically based renderer [42], and ran all our experiments on an Intel server equipped with four Xeon X7560 eight-core CPUs.

Precomputation. When precomputing the transfer matrices required by MFT, for each material, we pick a resolution such that every exemplar block has around 1000 non-empty voxels and 500 patches. As previously mentioned, the choice of this resolution does not depend on the actual data resolution of the underlying volume. In our case, each cloth block (representing one yarn crossing) contains about one million data voxels, and the final volumes in Figures 6.14 and 6.15 contain 2.5×10^{11} and 1.4×10^{12} effective voxels, respectively. Precomputation of a block takes roughly 4 hours, and storing the transfer matrices (as OpenEXR images) takes about 20MB. We distributed the precomputation tasks to Amazon Elastic Compute Cloud (EC2) by using between 25 and 80 `c1.xlarge` instances, where processing each exemplar block costs 2 USD. Note that the matrices are purely determined by material properties. Thus, after a one-time precomputation, we can use MFT to accelerate renderings of the material under any lighting condition. Also, we use the same precomputation, which was performed over rectangular blocks, when the volume is warped into draped shapes with shell-mapping (Figures 6.1, 6.14 and 6.15). Furthermore, one set of exemplar blocks may be used to synthesize models with very different structures (such as the cloth with different weave patterns in Figure 6.1 and both rows of Figure 6.15), amortizing the precomputation cost over a very large set of designs.



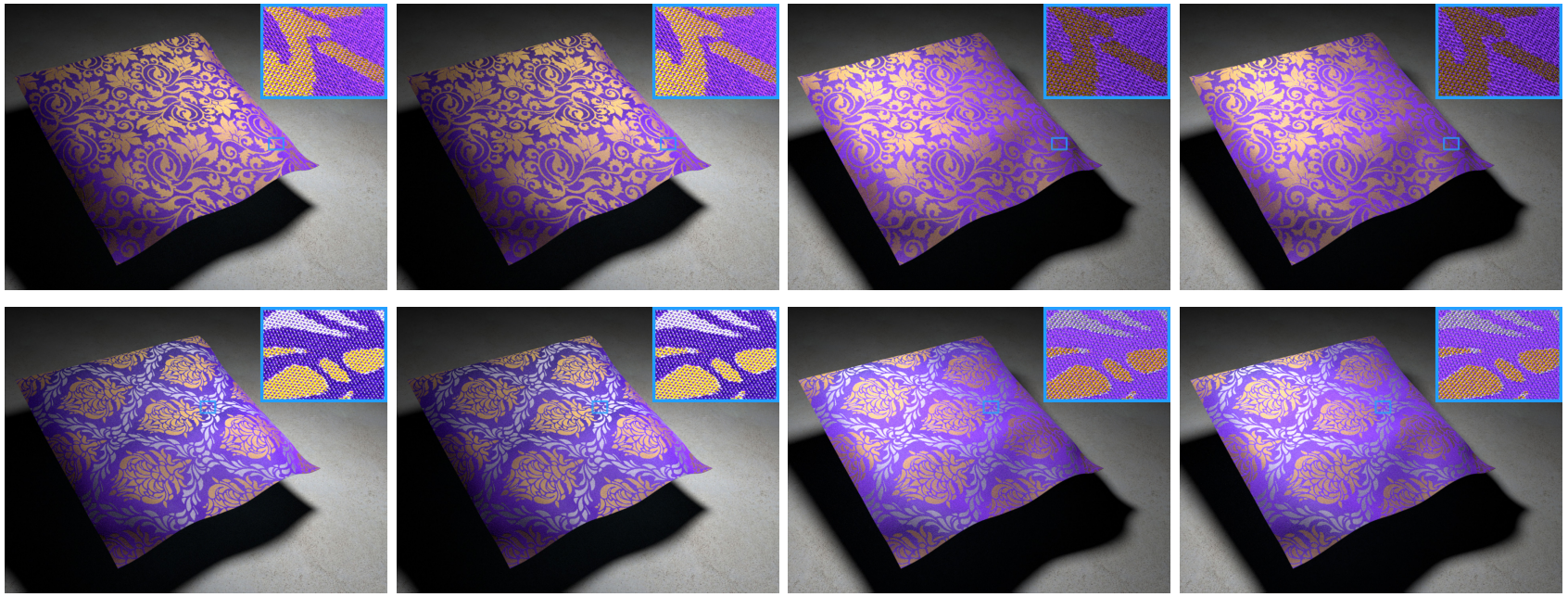
MFT path tracing path-traced reference

Figure 6.14: **Rendered fabrics in draped configurations:** (top) felt; (middle) twill weave; (bottom) velvet. The left column shows results rendered by our method (MFT). The center column shows two path-traced results: the left half of each image is rendered with fewer paths, sampled to achieve similar rendering time, but therefore, exhibits higher noise; the right half is rendered with the same number of samples but terminating all paths after 6 scatterings, showing significant darkening because of the lack of high-order scatterings. The right column shows path-traced references computed in much longer time. See Table 6.2 for performance numbers.

Single-colored fabrics. We first show rendered results for three types of fabrics (Figure 6.14) where the single-scattering albedo for each material stays constant (namely all fibers have identical colors) and equals around 0.975 (for the brightest color channel). Our method achieved respectively $10.3\times$, $9.0\times$, and $12.8\times$ speedup for these materials. The top row contains renderings for felt, a thick non-woven fabric with layers of disorganized fibers. The middle row shows those for a twill fabric, conveying characteristic diagonal lines. Unlike felt, this fabric contains well aligned fibers in two perpendicular directions which create strong anisotropic highlights. The bottom row exhibits rendered images of velvet with a visible surface composed of fibers sticking up from the base material, creating a distinctive grazing-angle highlight. Each of the three results has 25 precomputed exemplar blocks which can be reused to synthesize the same kind of material in arbitrary sizes.

Fabrics with complex designs. In addition, we created an exemplar database with 120 blocks using the method from [112]. With such a database, fabrics with complicated weave patterns can be constructed through synthesis. Since we precompute the transfer matrices for each exemplar block, the precomputation can be reused to render any synthesized cloth. Figures 6.1, 6.15 show renderings with three designs generated with this single database where our approach speeds up the renderings by $7.2\times$.

More information for images in Figures 6.14, 6.15 and 6.16 are available in Table 6.2. The results show that path tracing suffers from very long light paths because of the volume complexity and high albedo values. On the other hand, using the MFT method, we terminate the path after 6 scatterings, which bounds the maximal path length and yields significant speedups.



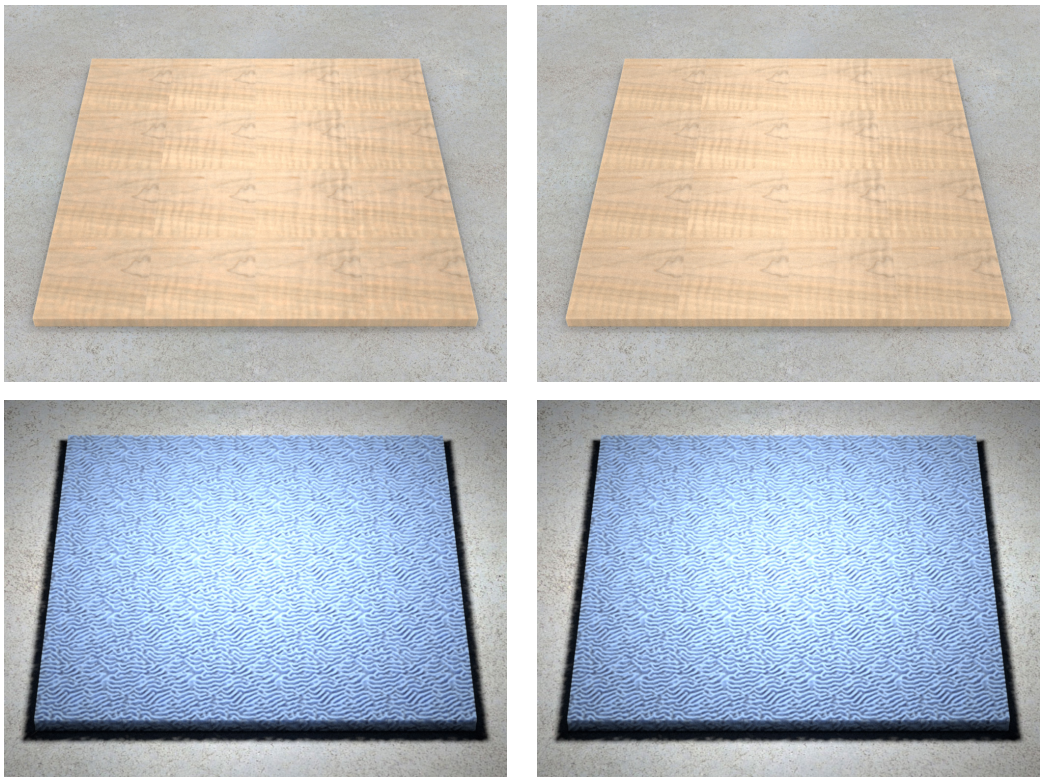
MFT path-traced reference MFT path-traced reference

Figure 6.15: **Fabrics with two designs** (both with 900×1500 blocks) rendered under two lighting configurations. All results rendered with our technique use the same set of precomputed transfer matrices. Performance information is in Table 6.2.

Scene	Total Blocks	Exemplar Blocks	Precomp. Time	Path Length		Time Cost		Speed Up
				PT	MFT	PT	MFT	
Felt	250 000	25	100	44.1	7.4	144	14	$10.3\times$
Twill	250 000	25	100	37.1	7.2	90	10	$9.0\times$
Velvet	250 000	25	100	71.0	7.9	192	15	$12.8\times$
Damask	1 350 000	120	480	65.8	7.2	108	15	$7.2\times$
Wood	256	25	100	17.9	6.3	17	5.6	$3.0\times$
Synthetic	625	25	100	23.6	6.9	6.6	1.6	$4.1\times$

Table 6.2: **Scene statistics.** The table shows the number of blocks in the scene, the number of exemplar blocks, the precomputation time for all exemplars (in hours), average path length, and rendering time (in minutes) for path tracing (PT) and our method (MFT). Felt, twill, and velvet correspond to Figure 6.14; damask to both designs in Figure 6.15; wood and synthetic to Figure 6.16. The MFT rendering time includes the portion spent on computing the source flux $\mathbf{\Phi}^s$ (by tracing particles from light sources), which is less than 2 minutes for all our results. The Monte Carlo matrix inversion step takes less than 15% of the rendering time for all scenes.

Materials beyond cloth. Finally, our technique can also be applied to non-cloth materials, as long as they are formed using a small set of exemplar blocks. Figure 6.16 shows two such results. On the left, we show a piece of finished wood represented with a micro-flake volume based on the data from [68]. The rendered wood conveys characteristic anisotropic highlights. Our method captures all these effects while filling in high-order scatterings accurately. On the right, we show a synthetic volume with a fine, coral-like structure made of a solid texture provided by [54]. Our result matches the ground truth very well. Although these datasets contain only a few million effective voxels, and are much less complicated than the cloth volumes, our method still achieved $3-4.1\times$ speedup over path tracing.



MFT path-traced reference

Figure 6.16: **Renderings of materials beyond cloth** under different lightings: (top) finished wood; (bottom) synthetic volume. Please see Table 6.2 for more information.

6.7 Conclusion

In this chapter, we introduced a precomputation-based approach to accelerate renderings of very complex volumetric materials built from sets of exemplar blocks. These materials are slow to render using pure Monte Carlo techniques and do not easily allow for diffusion approximations. Our algorithm separates low-order and high-order scattering and approximates the latter using a modular flux transfer framework. Based on the observation that introducing diffuser and isotropy events to long-enough light paths has little effect on accuracy, we showed that those paths can be split into precomputed and runtime components. The former can be evaluated by precomputing voxel-to-voxel, voxel-to-patch,

and patch-to-patch transfers for each exemplar block. The precomputation cost is amortized over many different lightings, shapes, and designs. An important component of our solution is a Monte Carlo matrix inversion method to solve the transfer problem with minimal storage cost; this means that our method has similar scalability to path tracing, but effectively traces much shorter paths for the same quality. Our results demonstrate a speed-up of more than an order of magnitude for thick cloths. In addition, the method can be used for non-cloth materials. We believe our algorithm could be directly used for high-quality rendering in interior design or movie production.

One limitation of this work is that we require a new precomputation if the exemplar blocks change their optical properties, such as single-scattering albedo. In the future, we plan to extend our framework to permit precomputation with material editing. Our Monte Carlo particle tracing based precomputation stage is quite expensive, thus further optimizations would be very useful. We would also like to explore heuristics for choosing the value of k adaptively, such as ones based on a frequency analysis of the resulting error. Combining our approach with Lightcuts [97] or other algorithms may be an interesting direction. Finally, we would like to push our approach to an even larger scale, for example, rendering a crowd of clothed characters.

CHAPTER 7

HIGH-ORDER SIMILARITY RELATIONS IN RADIATIVE TRANSFER

In this chapter, we switch gears and focus on radiative transfer theory, the mathematical rules governing how light transports in translucent media (including cloth).

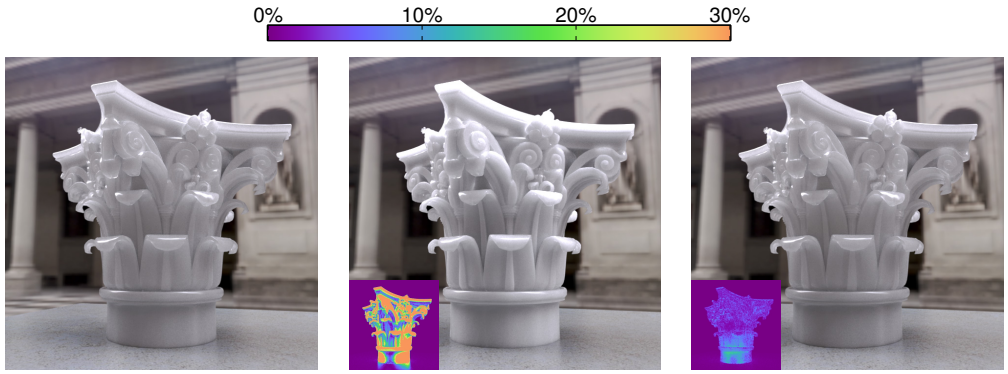
We introduce to computer graphics similarity theory, which provides fundamental insights into the structure of the parameter space of the radiative transfer equation (2.1). In addition, we develop a new algorithm to utilize this theory in its general high-order form. This work has been published at ACM SIGGRAPH 2014 [113].

7.1 Introduction

Many real-world materials including marble, jade, and human skin exhibit distinctive appearances arising from subsurface scattering of light. Understanding, simulating, and measuring this phenomenon has been an active research area in computer graphics for decades.

The physics of subsurface scattering is normally modeled with the radiative transfer framework [7]. The core of this framework is the radiative transfer equation (RTE) which governs how frequently light scatters and how it gets redirected or absorbed (when scattering occurs) via a set of scattering parameters.

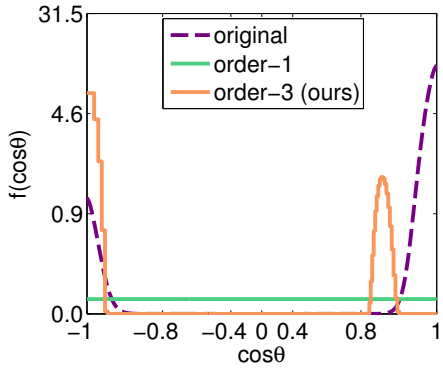
The parameter space of the RTE contains infinitely many equivalence classes such that different parameters in each class lead to identical solution radiance fields, under the assumption that these radiance fields have bounded angular frequencies. Similarity theory, introduced to applied physics by Wyman et al. [105],



(a) Reference
426.0 mins

(b) Order-1
119.1 mins (3.6X)

(c) Ours (Order-3)
115.2 mins (3.7X)



(d) Phase Function Plots

Figure 7.1: We introduce a new approach utilizing high-order similarity relations, which can be used to accelerate Monte Carlo rendering of translucent materials. (a) Reference path-traced rendering of a Corinthian capital made of a translucent material with a complicated phase function. (b) Image rendered with the same algorithm but using a reduced scattering coefficient and an isotropic phase function: although a 3.6X speedup is obtained, the resulting accuracy is poor (see the included relative error visualization with the color mapping shown above the renderings). (c) Image rendered using the same reduced scattering coefficient as (b) and a phase function provided by our method: with a slightly higher speedup, significantly better accuracy is obtained. (d) Plots of the phase functions used in (a, b, c). Our theory permits finding a tabulated function (the orange curve) accurately reproducing the reference appearance.

studies this property by deriving a hierarchy of equivalence relations called "similarity relations". Higher-order similarity relations offer finer partitions of the space, and parameters in the resulting equivalence classes can produce identical radiance with higher frequencies.

Previously, only a special case of the simplest order-1 similarity relation has been used in computer graphics. Furthermore, given a set of scattering parameters, computing an altered set adhering to (higher-order) similarity relations remains a challenge, especially for the altered phase function which is not uniquely determined by the relations.

In this chapter, we present a complete exposition of similarity theory and introduce practical algorithms to utilize this theory in its general higher-order forms. Our theoretical contributions include:

- Introducing to graphics the full derivation of similarity relations (Section 7.3.1) and discussing their connection to diffusion theory (Section 7.3.2).
- Developing novel algorithms to determine the existence of and to solve for the parameters (including absorption/scattering coefficients and a tabulated phase function) satisfying similarity relation of any given order (Section 7.4).

Our theory can lead to practical applications in forward and inverse rendering of translucent media. The presence of equivalence classes is a significant challenge in inverse rendering. This is because different parameters can provide very similar appearances, causing the optimization problem to be ill-conditioned. Section 7.5 introduces a proof-of-concept method reparameterizing the search space, so that gradient descent algorithms become much more effective in the new space.

Forward rendering is our main application (Section 7.6). We develop a simple procedure that takes a set of scattering parameters and outputs an altered set in a few seconds. Replacing the original parameters with the altered ones can

accelerate Monte Carlo rendering of optically dense and forward-scattering media (over 3X speedups can be achieved for volume path tracing). A key benefit offered by our method is that no changes need to be made to core rendering algorithms: only material scattering parameters, which are inputs to the renderer, are modified.

7.2 Overview

In this section, we first briefly revisit the basic concept of radiative transfer and present a mathematical description of similarity theory. Then, we describe the computational challenges for forward and inverse rendering of translucent media and our plan to tackle these challenges.

Radiative transfer. The radiative transfer equation (2.1) describes that the directional derivative of the radiance field L is determined by its value via the *extinction coefficient* σ_t, an integral of L at the same location via the *scattering coefficient* σ_s and the *phase function* f, and the *source term* Q. In addition, $\sigma_t = \sigma_a + \sigma_s$ where σ_a is called the *absorption coefficient*. In the rest of this chapter, we assume that the media is isotropic and has no self-emission. This causes f to become a 1D function of $(\boldsymbol{\omega} \cdot \boldsymbol{\omega}')$ and Q to vanish, yielding

$$(\boldsymbol{\omega} \cdot \nabla) L(\boldsymbol{\omega}) = -\sigma_t L(\boldsymbol{\omega}) + \sigma_s \int_{\mathbb{S}^2} f(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) L(\boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}'. \tag{7.1}$$

Note that anisotropic media [43] require f to be full 4D functions and are beyond the scope of this chapter. In (7.1) and the rest of this chapter, we use $h(\cdot)$ to indicate that h is a 1D function defined on $[-1, 1]$ (for any h).

Similarity theory. Theoretically, assuming the solution radiance L in the RTE (7.1) is band-limited (in the spherical harmonics, or SH, domain), there exist altered parameters σ_t^*, σ_s^*, and $f^*(\cdot)$, such that the corresponding altered RTE

$$(\boldsymbol{\omega} \cdot \nabla) L(\boldsymbol{\omega}) = -\sigma_t^* L(\boldsymbol{\omega}) + \sigma_s^* \int_{\mathbb{S}^2} f^*(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) L(\boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}' \tag{7.2}$$

has a solution which equals that of (7.1) exactly. Similarity theory [105] describes the relations between the altered parameters and the original ones, which are presented in Section 7.3. Based on these relations, the parameter space can be partitioned into multiple equivalence classes. In practice, when the assumption does not hold perfectly, parameters in one equivalence class produce approximately identical appearances.

Inverse rendering: challenges. Given the complicated and highly non-linear relation between the scattering parameters and the resulting appearance, inverse rendering is usually modeled as an optimization problem where the parameter space needs to be explored locally [99, 35, 16, 31]. This process, however, can be extremely expensive as the search space is often high-dimensional, and exploring it requires iteratively solving the forward problem, which is challenging by itself. The presence of the equivalence classes makes the inverse problem even more difficult as it creates ambiguities between different sets of parameters which can cause the optimization to be ill-conditioned.

Inverse rendering: our approach. Similarity theory suggests the locations in the parameter space where the ambiguities occur. Section 7.3.3 illustrates such an example using a simple 2D space. Based on this understanding, we introduce a proof-of-concept method (Section 7.5) which warps the space in a non-linear manner so that it becomes much easier for gradient based methods to find good

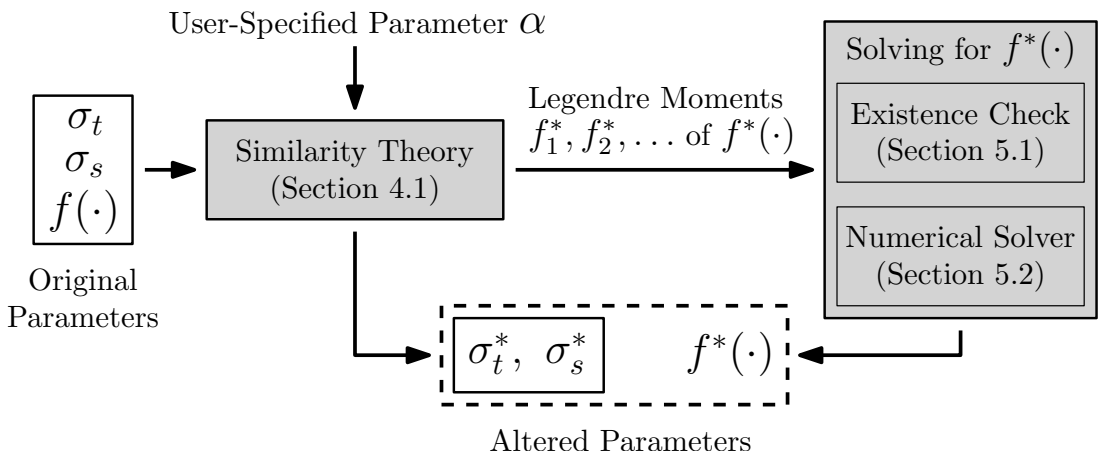


Figure 7.2: **Our pipeline** to speedup forward rendering of translucent media. It takes the original scattering parameters as well as a user-specified $\alpha \in (0, 1)$ and outputs the altered parameters.

solutions.

Forward rendering: challenges. Forward rendering of translucent materials is our main application. It requires solving (7.1) which in general has no closed-form solution, and accurate numerical solutions often involve Monte Carlo simulations. To render a normal sized object made of optically dense materials, such as milk and marble, hundreds or thousands of subsurface scatterings need to be simulated on each light path, yielding slow performance. Among such materials, the highly forward-scattering ones are particularly difficult to handle. They include phase functions that send light into very concentrated regions, causing path tracing based algorithms to produce high noise, and photon mapping or many-lights methods to require a massive number of photons or virtual lights to avoid intense artifacts or energy loss.

Forward rendering: our approach. We tackle the challenge of rendering optically dense and forward-scattering materials using similarity theory. Particularly,

we look for equivalent parameters σ_t^*, σ_s^*, $f^*(\cdot)$ with $\sigma_t^* < \sigma_t$ because a smaller extinction coefficient means fewer scattering events to simulate and less computation required. The most basic version of this idea, which reduces σ_s and sets $f^*(\cdot)$ to isotropic, has been used in graphics but can result in poor accuracy (Section 7.3.2).

Figure 7.2 previews the pipeline of our method. The user provides the original scattering parameters σ_t, σ_s, $f(\cdot)$ as well as an extra parameter[1] $\alpha \in (0, 1)$ controling the tradeoff between performance and accuracy. The first component of our pipeline computes the altered absorption and scattering coefficients based on similarity theory (Section 7.3.1). The altered phase function $f^*(\cdot)$, however, is not directly given. Instead, similarity theory specifies the desired Legendre moments $f_1^*, f_2^*, \ldots$ of $f^*(\cdot)$. The second component of the pipeline then numerically solves for $f^*(\cdot)$ as a tabulated function given those moments (Section 7.4).

Our approach is easy to implement (pseudocode is in Section 7.6.1) and straight-forward to use: the user can simply replace the original scattering parameters with the altered ones (which are the outputs of our pipeline). The base rendering method does not need to be changed. Thorough experimental evaluations of our method are in Section 7.7.

7.3 Similarity theory

Similarity theory was originally introduced to applied physics by Wyman et al. [105, 106]. It studies the equivalence classes of the RTE's parameter space by introducing a hierarchy of equivalence relations called *similarity relations*.

[1] In this chapter, we use α to denote the user-specified parameter (which equals the ratio between the altered and the original scattering coefficients, as described in Section 7.4). In the previous chapters, this symbol has been used to indicate the single scattering albedo σ_s/σ_t.

Note that, given the original scattering parameters, the similarity relations do not directly provide the values of all altered parameters, and computing these values (the altered phase function in particular) is non-trivial. We introduce a novel approach to solve for these parameters in Section 7.4.

In this section, we first present the full derivation of simialrity relations (Section 7.3.1) following the original version proposed by Wyman et al. [105]. Then, Section 7.3.2 discusses the connection between similarity theory and approaches based on first-order approximations of the RTE (such as diffusion methods). Finally, Section 7.3.3 shows an example of capturing the structure of a simple RTE's parameter space using the derived relations.

7.3.1 Derivation of similarity relations

We present the derivation (following [105]) of a set of relations between the original parameters σ_a, σ_s, $f(\cdot)$ and the altered ones σ_a^*, σ_s^*, $f^*(\cdot)$ such that the original RTE (7.1) and its altered version (7.2) have identical solution radiance L, based on the assumption that L has bounded directional frequency. The resulting relations are in (7.18).

Rearranging the terms in the original RTE (7.1) yields

$$(\boldsymbol{\omega} \cdot \nabla) L(\boldsymbol{\omega}) + I(\boldsymbol{\omega}) = 0 \tag{7.3}$$

where

$$I(\boldsymbol{\omega}) := \sigma_t L(\boldsymbol{\omega}) - \sigma_s \int_{\mathbb{S}^2} f(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) L(\boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}'.$$

Similarly, the altered RTE (7.2) can be rewritten as

$$(\boldsymbol{\omega} \cdot \nabla) L(\boldsymbol{\omega}) + I^*(\boldsymbol{\omega}) = 0. \tag{7.4}$$

Then having one solution L satisfying both (7.3) and (7.4) implies that for all $\boldsymbol{\omega} \in \mathbb{S}^2$,

$$I(\boldsymbol{\omega}) = I^*(\boldsymbol{\omega}). \tag{7.5}$$

To derive the similarity relations from (7.5), one can represent $I(\boldsymbol{\omega})$ and $I^*(\boldsymbol{\omega})$ in SH and equate the corresponding SH coefficients. To write $I(\boldsymbol{\omega})$ in SH, the radiance field L and the phase function $f(\cdot)$ need to be expanded, yielding

$$L(\boldsymbol{\omega}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} a_{mn} Y_n^m(\boldsymbol{\omega}), \tag{7.6}$$

$$\begin{aligned} f(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) &= \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} f_n P_n(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) \\ &= \sum_{n=0}^{\infty} \sum_{m=-n}^{n} f_n Y_n^m(\boldsymbol{\omega}) \bar{Y}_n^m(\boldsymbol{\omega}') \end{aligned} \tag{7.7}$$

where Y_n^m is the SH basis function, P_n is the Legendre polynomial of degree n,

$$f_n = 2\pi \int_{-1}^{1} f(t) P_n(t) \, \mathrm{d}t \tag{7.8}$$

is the n-th *Legendre moment* of $f(\cdot)$, and the bar superscript denotes complex conjugation. The second equality in (7.7) follows the Addition Theorem [3]. For heterogeneous materials, both a_{mn} and f_n have spatial dependencies.

Given (7.6) and (7.7), it holds that

$$
\begin{aligned}
I(\boldsymbol{\omega}) = & \sum_{n,m} \sigma_t a_{mn} Y_n^m(\boldsymbol{\omega}) - \\
& \int_{\mathbb{S}^2} \left[\sum_{n,m} a_{mn} Y_n^m(\boldsymbol{\omega}') \right] \left[\sum_{i,j} \sigma_s f_i Y_i^j(\boldsymbol{\omega}) \bar{Y}_i^j(\boldsymbol{\omega}') \right] \mathrm{d}\boldsymbol{\omega}' \\
= & \sum_{n,m} \sigma_t a_{mn} Y_n^m(\boldsymbol{\omega}) - \\
& \sum_{n,m} \sum_{i,j} \left(a_{mn} Y_i^j(\boldsymbol{\omega}) \sigma_s f_i \underbrace{\int_{\mathbb{S}^2} Y_n^m(\boldsymbol{\omega}') \bar{Y}_i^j(\boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}'}_{= \delta_{ni} \, \delta_{mj}} \right) \\
= & \sum_{n,m} \sigma_t a_{mn} Y_n^m(\boldsymbol{\omega}) - \sum_{n,m} a_{mn} Y_n^m(\boldsymbol{\omega}) \sigma_s f_n \\
= & \sum_{n,m} a_{mn} \sigma_{tr,n} Y_n^m(\boldsymbol{\omega})
\end{aligned}
$$

where δ_{ij} is the Kronecker delta which equals 1 if $i = j$ and 0 otherwise. Then,

$$
I(\boldsymbol{\omega}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} a_{mn} \sigma_{tr,n} Y_n^m(\boldsymbol{\omega}), \tag{7.9}
$$

$$
I^*(\boldsymbol{\omega}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} a_{mn} \sigma_{tr,n}^* Y_n^m(\boldsymbol{\omega}). \tag{7.10}
$$

where

$$
\begin{aligned}
\sigma_{tr,n} &:= \sigma_t - \sigma_s f_n = \sigma_a + \sigma_s (1 - f_n), \\
\sigma_{tr,n}^* &:= \sigma_t^* - \sigma_s^* f_n^* = \sigma_a^* + \sigma_s^* (1 - f_n^*)
\end{aligned} \tag{7.11}
$$

are called the *reduced extinction coefficients* of order n. Note that a_{mn} appears in both (7.9) and (7.10) since $I(\boldsymbol{\omega})$ and $I^*(\boldsymbol{\omega})$ are assumed to share the same radiance field L.

From (7.9) and (7.10), the SH coefficients of $I(\boldsymbol{\omega})$ and $I^*(\boldsymbol{\omega})$ can now be equated, which leads to $a_{mn}\sigma_{tr,n} = a_{mn}\sigma_{tr,n}^*$ for all $n \geq 0$ and $-n \leq m \leq n$.

Namely,

$$a_{mn}(\sigma_{tr,n} - \sigma^*_{tr,n}) = \\ a_{mn}[(\sigma_a - \sigma^*_a) + (\sigma_s(1 - f_n) - \sigma^*_s(1 - f^*_n))] = 0. \quad (7.12)$$

Since (7.12) needs to hold for all n and m, consider a special case where $n = m = 0$. It holds that

$$a_{00} = \int_{\mathbb{S}^2} Y^0_0(\boldsymbol{\omega}) L(\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega} = \frac{\phi}{2\sqrt{\pi}} \quad (7.13)$$

where $\phi := \int_{\mathbb{S}^2} L(\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega}$ is the *fluence*. Because $f(\boldsymbol{\omega}' \cdot \boldsymbol{\omega})$ and $f^*(\boldsymbol{\omega}' \cdot \boldsymbol{\omega})$, as functions of $\boldsymbol{\omega}'$, are probability densities over $\mathbb{S}^2$, it holds that $f_0 = f^*_0 = 1$. Then, (7.12) becomes $\frac{\phi}{2\sqrt{\pi}}(\sigma_a - \sigma^*_a) = 0$. Because ϕ is generally non-zero, this implies

$$\sigma_a = \sigma^*_a. \quad (7.14)$$

In general, for $n \geq 1$, given (7.14), (7.12) becomes

$$a_{mn}[\sigma_s(1 - f_n) - \sigma^*_s(1 - f^*_n)] = 0. \quad (7.15)$$

To ensure that (7.15) holds for any L (namely for arbitrary a_{mn}), we need to have

$$\sigma_s(1 - f_n) = \sigma^*_s(1 - f^*_n). \quad (7.16)$$

Similarity relations. Wyman et al. [105] showed that the only solution adhering to (7.14) and (7.16) for all n, m is the trivial one: $\sigma^*_a = \sigma_a$, $\sigma^*_s = \sigma_s$, and $f^*(\cdot) \equiv f(\cdot)$. Thus, in general, there is no "perfect" similarity relation. However, when L is band-limited in SH domain, we have

$$a_{mn} = 0 \quad \text{for} \quad n > N, \; -n \leq m \leq n \quad (7.17)$$

where N is a constant capturing the maximal angular frequency. If $N = 1$, for example, L is called *linearly anisotropic*. When (7.17) holds, (7.16) only needs to be enforced for $1 \leq n \leq N$, yielding the ***similarity relation of order*** N:

$$\begin{aligned} \sigma_a &= \sigma_a^*, \\ \sigma_s(1 - f_n) &= \sigma_s^*(1 - f_n^*) \quad \text{for} \quad 1 \leq n \leq N. \end{aligned} \tag{7.18}$$

In practice, (7.17) may not hold everywhere inside the medium. In this case, the altered RTE (7.2) will have a solution radiance approximating that of the original (7.1).

The Legendre moment constraints in lower-order similarity relations are subsets of those in higher-order ones. Therefore, (7.18) essentially provides *a hierarchy of equivalence relations* that partition the parameter space of a RTE with different granularity.

Spatial dependency. Similarity relation (7.18) needs to be satisfied at every location $\mathbf{x}$ (which has been dropped for notational convenience) within the material volume. Namely, for heterogeneous materials where σ_s, σ_a, and $f(\cdot)$ are spatially varying, the altered parameters σ_s^*, σ_a^*, and $f^*(\cdot)$ should also have spatial dependencies so that (7.18) is satisfied independently for each $\mathbf{x}$.

Generalized forms. The similarity relations (7.18) force the absorption coefficient σ_a to remain unchanged. When $\sigma_a^* \neq \sigma_a$, the solution radiance L to the original (7.1) and the altered RTE (7.2) are normally not identical. Wyman et al. [106] showed that if one weakens the requirement of identical L and only asks for equal resulting fluence $\phi = \int_{\mathbb{S}^2} L(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega}$, generalizations of (7.18) can be obtained. In particular, the *generalized order-1 and order-2 similarity relations* are

respectively

$$\sigma_a \sigma_{tr,1} = \sigma_a^* \sigma_{tr,1}^*, \qquad \frac{\sigma_a \sigma_{tr,1} \sigma_{tr,2}}{\sigma_s(1-f_2)} = \frac{\sigma_a^* \sigma_{tr,1}^* \sigma_{tr,2}^*}{\sigma_s^*(1-f_2^*)}. \tag{7.19}$$

When $\sigma_a^* = \sigma_a$, (7.19) reduces to (7.18) with $N = 1$ and $N = 2$.

Since most graphics applications care about radiance L (which determines an object's appearance) instead of fluence ϕ, we focus on the standard similarity relations (7.18) in the rest of this chapter. Please refer to Section A.2 for more information on the generalized versions.

7.3.2 Discussion: relation to first-order methods

In many prior works [8, 25], it was common to set

$$\sigma_a^* = \sigma_a, \quad \sigma_s^* = \sigma_s(1-f_1), \quad f^*(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) = \tfrac{1}{4\pi}. \tag{7.20}$$

It is easy to verify that the altered phase function $f^*(\cdot)$ has $f_1^* = 0$. Thus, $\sigma_s^*(1 - f_1^*) = \sigma_s^* = \sigma_s(1-f_1)$, and (7.20) satisfies the order-1 similarity relation.

The altered parameters in (7.20) are also used by diffusion methods [47, 2, 15] where σ_s^* is called the *reduced scattering coefficient*. In fact, the order-1 similarity relation and the diffusion approximation share the same assumption that L is linearly anisotropic (namely (7.17) holds with $N = 1$), so they offer similar levels of accuracy. We present an alternative derivation of the diffusion equation (DE) using this assumption in Section A.1.

On the other hand, methods based on first-order approximations, including (7.20), have limited accuracy. These methods assume that the radiance field is linearly anisotropic and can perform poorly in optically thin or close-to-boundary regions where the assumption is often violated. Furthermore, phase functions

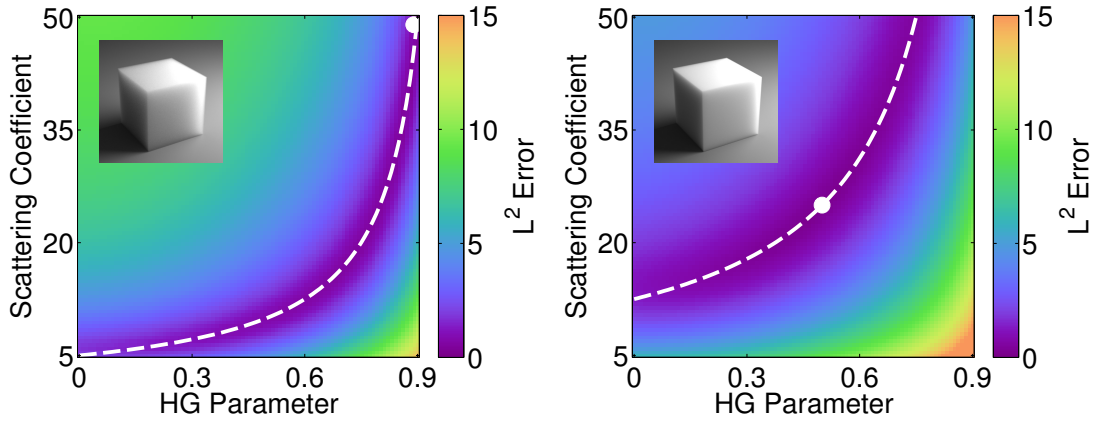


Figure 7.3: **Equivalence classes of a 2D parameter space.** White dots indicate the reference parameter points: $(0.9, 50)$ for the left plot and $(0.5, 25)$ for the right. Dashed lines contain all points belonging to the same equivalence class (defined by the order-1 similarity relation) as the references. Low-error regions on the error surfaces (in false color) match the predicted equivalence classes, confirming the theory.

with similar first moments can lead to dramatically different appearances (examples are shown in Figures 7.1 and 7.9 as well as by Gkioulekas et al. [29]). Unfortunately, first-order methods care only about the first moment of a phase function and cannot capture all these varying appearances, as demonstrated in Sections 7.7.2 and 7.7.3.

To our knowledge, there is no prior work in computer graphics which considers higher-order similarity relations. In Section 7.4, we show how to solve for the altered parameters satisfying the similarity relation of any given order.

7.3.3 Example: equivalence classes

We now illustrate the structure of the parameter space of a simple RTE and discuss how such structure can be exploited to benefit forward and inverse rendering of volumetric media.

Consider a RTE with a fixed absorption coefficient and a Henyey-Greenstein (HG) phase function [37]. Then its parameter space is 2D: one for the scattering coefficient σ_s and the other for the HG parameter g. Because the first Legendre moment of an HG function with parameter g is simply g itself, two parameter points (g, σ_s) and (g', σ_s') belong to the same equivalence class defined by the order-1 similarity relation[2] if $\sigma_s(1-g) = \sigma_s'(1-g')$. Figure 7.3 shows the equivalence classes of two reference parameters $(0.9, 50)$ and $(0.5, 25)$ plotted as dashed lines.

To validate the equivalence classes defined by similarity theory, we created multiple renderings of a homogeneous, unit-sized cube (under side lighting) using different scattering parameters. Denote the image rendered with parameters g and σ_s as $\mathbf{I}(g, \sigma_s)$. The insets in Figure 7.3 show images rendered with the reference parameters. For each point (g, σ_s) in the space, define an error function

$$d(g, \sigma_s) := \|\mathbf{I}(g, \sigma_s) - \mathbf{I}_0\|_2 \tag{7.21}$$

where $\mathbf{I}_0$ is the image rendered with the reference parameters. The error surfaces defined by d are visualized as plot backgrounds. We can see that the shapes of low-error regions match the curves predicted by similarity theory well. Note that the error values over the dashed lines are not exactly zero, since the equivalence classes are defined based on the assumption that the radiance is linearly anisotropic, which is normally not the case near boundaries.

Inverse rendering. The presence of these low-error regions causes inverse rendering to be very challenging. One reason is that the shapes of these regions are not convex, so that many optimization algorithms are not guaranteed to find

[2]Similarity relations beyond order-1 are not useful for this parameter space, as each equivalence class would contain only a single point.

a global optimum. Furthermore, within the low-error regions, the gradient is fairly small and can be easily dominated by Monte Carlo or measurement noise. Based on this understanding, Section 7.5 presents a simple method that reparameterizes the search space, causing gradient based methods to be much more effective.

Forward rendering. We can exploit the structure of the parameter space to benefit forward rendering applications (Section 7.6). To render an object with scattering parameters coming from the upper-right region of Figure 7.3, for instance, we can instead use a set of parameters in the same equivalence class but located at the bottom-left of the space, as both sets of parameters lead to approximately the same appearance. By picking a smaller scattering coefficient, the material's optical density is reduced, causing light to scatter less frequently. Consequently, fewer scattering events need to be simulated, and speedups can be obtained.

7.4 Solving for altered parameters

Deriving the similarity relations (7.18) is only half the story. Applications such as forward rendering require full sets of parameters: the values of σ_a^*, σ_s^*, and a complete phase function $f^*(\cdot)$. Unfortunately, only $\sigma_a^* = \sigma_a$ is given directly by (7.18). The other relations, $\sigma_s(1 - f_n) = \sigma_s^*(1 - f_n^*)$, are constraints.

To determine σ_s^*, we consider the ratio between σ_s^* and σ_s:

$$\alpha := \sigma_s^*/\sigma_s. \tag{7.22}$$

In our forward rendering pipeline (Figure 7.2), this ratio is selected by the user.

Given α, we have $\sigma_s^* = \alpha\sigma_s$, and the only parameter that remains unknown is the altered phase function $f^*(\cdot)$. For fixed order N, the Legendre moments of $f^*(\cdot)$ need to satisfy

$$f_0^* = 1, \quad f_i^* = 1 - \frac{1 - f_i}{\alpha} \text{ for } 1 \leq i \leq N. \tag{7.23}$$

Unfortunately, computing $f^*(\cdot)$ given $f_0^*, \ldots, f_N^*$ is non-trivial. As a phase function, $f^*(\cdot)$ needs to be *nonnegative*. Discarding all moments higher than order-N by setting $f^*(t) = \sum_{n=0}^{N} \frac{2n+1}{4\pi} f_n^* P_n(t)$, however, generally does not offer nonnegativity. In addition, given the Legendre moment constraints, a nonnegative $f^*(\cdot)$ may not exist at all.

Wyman et al. [105] proposed a simple approach to provide $f^*(\cdot)$. This method does not allow N and α to be specified simultaneously. Instead, it takes N as the user input and constructs a phase function $f^*(\cdot)$ with $f_N^* = 0$. This method, therefore, offers insufficient flexibility: in many applications including forward rendering, we need to control both N and α to achieve good performance and accuracy.

In this section, we introduce a general technique to find $f^*(\cdot)$ for any given N and α. Section 7.4.1 presents existence conditions of $f^*(\cdot)$ given $f_0^*, \ldots, f_N^*$. Section 7.4.2 introduces an algorithm to solve for $f^*(\cdot)$ numerically as a tabulated (piecewise-constant) function.

7.4.1 Existence of the altered phase function

We now show the sufficient and necessary conditions for the existence of a nonnegative function $f^*(\cdot)$ with its Legendre moments $f_0^*, \ldots, f_N^*$ given.

For $f^*(\cdot)$, its n-th *monomial moment* is $\gamma_n^* := \int_{-1}^{1} f(t)t^n \, \mathrm{d}t$. Since $P_n(\cdot)$ is a poly-

nomial of degree n, the Legendre moments $f_0^*, \ldots, f_N^*$ and monomial moments $\gamma_0^*, \ldots, \gamma_N^*$ of $f^*(\cdot)$ uniquely determine each other. Given these Legendre moments, the corresponding monomial moments can be computed by solving a linear system (see Section A.3 for details). It follows that determining whether $f^*(\cdot)$ exists given its Legendre moments $f_0^*, \ldots, f_N^*$ is equivalent to checking its existence given the monomial moments $\gamma_0^*, \ldots, \gamma_N^*$. The latter is called the *truncated Hausdorff moment problem* and has been studied in probability theory [12]. In fact, $f^*(\cdot)$ exists if and only if certain Hankel matrices formed using the monomial moments are positive semi-definite. The following theorem provides a formal description of this result (see Theorems 4.1 and 4.3 in Curto and Fialkow's work [12] for the proof).

Theorem 1. *Given $\gamma_0^*, \gamma_1^*, \ldots, \gamma_N^*$ with $\gamma_0^* > 0$. For each n, let $\mathbf{U}_n$, $\mathbf{V}_n$, and $\mathbf{W}_n$ be $n \times n$ Hankel matrices such that*

$$U_n(i,j) = \gamma_{i+j-2}^*,\ V_n(i,j) = \gamma_{i+j-1}^*,\ W_n(i,j) = \gamma_{i+j}^* \tag{7.24}$$

for $1 \leq i, j \leq n$. Then, a nonnegative function $f^*(\cdot)$ with monomial moments $\gamma_0^*, \gamma_1^*, \ldots, \gamma_N^*$ exists if and only if:

- *(odd case) when $N = 2k+1$,*

$$\mathbf{U}_{k+1} - \mathbf{V}_{k+1} \succcurlyeq 0, \quad \mathbf{U}_{k+1} + \mathbf{V}_{k+1} \succcurlyeq 0 \tag{7.25}$$

  where "$\succcurlyeq 0$" denotes positive semi-definiteness *of a matrix;*

- *(even case) when $N = 2k$,*

$$\mathbf{U}_{k+1} \succcurlyeq 0, \quad \mathbf{U}_k - \mathbf{W}_k \succcurlyeq 0. \tag{7.26}$$

Based on Theorem 1, a function IFEXISTS() can be easily implemented which takes the desired Legendre moments $f_0^*, \ldots, f_N^*$ and returns a Boolean indicating if $f^*(\cdot)$ exists.

7.4.2 Computing the altered phase function

Although Theorem 1 allows us to efficiently check the existence of the altered phase function $f^*(\cdot)$, it does not provide a practical way to find one (if it exists). Next, we introduce an algorithm to solve for $f^*(\cdot)$ numerically. The resulting $f^*(\cdot)$ can then be used for physically-based rendering applications (Section 7.6).

We represent $f^*(\cdot)$ as the linear combination of k basis functions $g_1(\cdot), \ldots, g_k(\cdot)$:

$$f^*(t) = \sum_{i=1}^{k} c_i \, g_i(t). \tag{7.27}$$

Note that this can lose generality as the bases may not be able to represent all nonnegative functions on $[-1, 1]$. The selection of k is discussed in the end of this subsection.

Given (7.27) and (7.8), the n-th Legendre moment of $f^*(\cdot)$ equals

$$f_n^* = 2\pi \int_{-1}^{1} \left(\sum_{i=1}^{k} c_i \, g_i(t) \right) P_n(t) \, \mathrm{d}t = \sum_{i=1}^{k} c_i g_{i,n} \tag{7.28}$$

where

$$g_{i,n} := 2\pi \int_{-1}^{1} g_i(t) P_n(t) \, \mathrm{d}t$$

is the n-th Legendre moment of $g_i(\cdot)$.

Let $\mathbf{f}^* := (f_0^* \; f_1^* \; \ldots \; f_N^*)^T$, $\mathbf{c} := (c_1 \; c_2 \; \ldots \; c_k)^T$, and

$$\mathbf{G} := \begin{pmatrix} g_{1,0} & g_{2,0} & \cdots & g_{k,0} \\ g_{1,1} & g_{2,1} & \cdots & g_{k,1} \\ \vdots & \vdots & \vdots & \vdots \\ g_{1,N} & g_{2,N} & \cdots & g_{k,N} \end{pmatrix},$$

then (7.28) with $n = 0, 1, \ldots, N$ can be rewritten as

$$\mathbf{f}^* = \mathbf{G}\,\mathbf{c}. \tag{7.29}$$

To summarize, if $g_1(\cdot), g_2(\cdot), \ldots, g_k(\cdot)$, which determine $\mathbf{G}$, are given, we need to solve for $\mathbf{c}$ such that $f^*(\cdot)$ is nonnegative and the Legendre moment constraints (7.29) are satisfied.

In our implementation, we pick the boxcar basis functions:

$$g_i(t) := \begin{cases} 1 & -1 + \frac{2i-2}{k} \leq t < -1 + \frac{2i}{k} \\ 0 & \text{otherwise} \end{cases}, \tag{7.30}$$

for $i = 1, 2, \ldots, k$. We choose this basis for its simplicity and flexibility. Under (7.30), $f^*(\cdot)$ becomes piecewise-constant, and its value in the i-th piece simply equals c_i. It follows that $f^*(\cdot)$ is nonnegative if and only if $\mathbf{c} \geq \mathbf{0}$ (defined component-wise). We then would like to find a nonnegative k-dimensional vector $\mathbf{c}$ satisfying (7.29). This system, however, is normally under-constrained since k, the number of pieces in $f^*(\cdot)$, can be much greater than $(N+1)$, the amount of Legendre moment constraints. Therefore, we regularize the system by introducing a smoothness term $\mathbf{Sc}$ with $\mathbf{S} \in \mathbb{R}^{(k-2)\times k}$ being a 1D Poisson matrix

$$\mathbf{S} = \begin{pmatrix} -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & & \cdots & & \\ & & & & -1 & 2 & -1 \end{pmatrix}.$$

This smoothness term captures the second derivative of $f^*(\cdot)$, and we would like $\|\mathbf{Sc}\|_2$ to be minimized. We choose the 2-norm since the solution $\mathbf{c}$ is robust to the choice of k, and the resulting phase function $f^*(\cdot)$ tends to send light into a wide range of directions, which is a desirable feature.

Let $\mathbf{Q} := \mathbf{S}^T\mathbf{S}$, then $\|\mathbf{Sc}\|_2^2 = \mathbf{c}^T\mathbf{Qc}$, and we need to find the minimizer to the following quadratic programming problem:

$$\min_{\mathbf{c}}(\mathbf{c}^T\mathbf{Qc}) \quad \text{subject to} \quad \mathbf{c} \geq \mathbf{0},\ \mathbf{G}\,\mathbf{c} = \mathbf{f}^*. \tag{7.31}$$

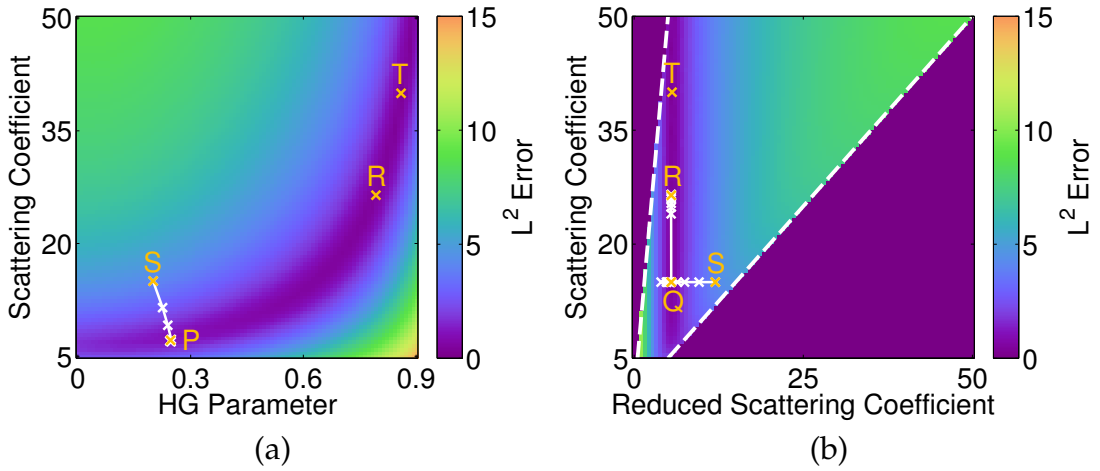


Figure 7.4: **Search spaces for an inverse rendering problem:** (a) the original space; (b) the reparameterized space. The plotted region in (a) maps to the area enclosed by the dashed lines in (b). Using the original space, the stochastic gradient descent (SGD) algorithm starting from point S is trapped at point P, which is far from the real solution T. Using the reparameterized space, the algorithm is able to find point R that is much closer to the real solution.

Since $\mathbf{Q}$ is positive semi-definite, the global optimum can be found in polynomial time [55]. We solve (7.31) using the Gurobi Optimization Libraries [33].

Selecting k. Given $\mathbf{f}^*$, if the existence condition (7.25) or (7.26) is violated, the quadratic programming problem in (7.31) is guaranteed to be infeasible. However, the converse is not necessarily true: when (7.31) is infeasible, there could still exist some $f^*(\cdot)$ which cannot be represented using the k basis functions $g_1(\cdot), g_2(\cdot), \ldots, g_k(\cdot)$. In this case, we need a larger set of bases. One possibility to determine the value of k is to start with some relatively small k_0 and double it whenever (7.31) is infeasible, but there should be a solution (according to Theorem 1). In practice, however, we found that simply setting $k = 360$ is sufficient to find the solutions in all our experiments.

7.5 Application: inverse rendering

In this section, we first describe an inverse rendering problem which we believe is a good example to demonstrate the practical usefulness of similarity theory. Then, we introduce a reparameterized search space in which gradient descent algorithms converge to a good solution much faster.

The problem. Consider the problem of acquiring the material parameters of a cube made of a scattering medium. Given a photograph of the cube lit by an area light from the side (identical to the setting used in Section 7.3.3), we assume that the cube has a HG phase function and its absorption coefficient σ_a is given. The goal is to find the scattering coefficient σ_s and HG parameter g.

Our solution. Given the highly complicated and non-linear relation between the parameters and the resulting rendered image, exact analytical solutions do not exist for this problem. Instead, we use the error function d defined in (7.21) with $\mathbf{I}_0$ set to the input image, and solve for g, σ_s such that the error is minimized. This optimization problem can be solved using the stochastic gradient descent (SGD) algorithm. The nondeterminism is caused by the fact that the gradients need to be obtained through Monte Carlo simulations and can be noisy.

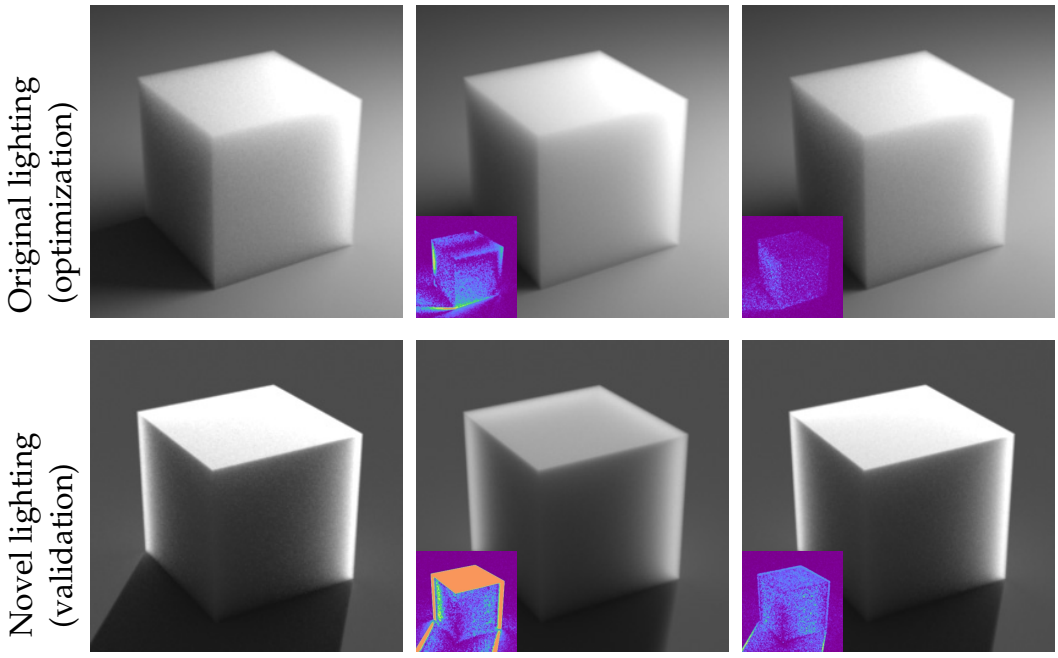
Unfortunately, the use of a relatively low-frequency (soft) lighting results in large regions where $d(g, \sigma_s)$ is close to zero (as shown in Figure 7.3), and the gradient values within these regions are very small and can be dominated by Monte Carlo or measurement noise. Consequently, after hitting this region, it becomes very difficult for SGD to make further progress. Figure 7.4-a shows such an example in which $\mathbf{I}_0$ is generated using parameters at point T. If we

choose point S as the initial guess and execute SGD, the solution point moves to the low-error region (indicated in purple) very quickly, but then gets "trapped" there. Figure 7.5-b demonstrates that the solution (point P) found by this process generalizes poorly to high-frequency (hard) lighting conditions.

To address this problem, we reparameterize the search space (Figure 7.4-b) by replacing the horizontal axis by the reduced scattering coefficient $\sigma_s' := (1-g)\sigma_s$. Under this reparameterization, both axes have the same units, and the error surface becomes significantly more regular (Figure 7.4-b). To search for the solution in the new space, we perform two one-dimensional SGDs. First, we fix σ_s and look for σ_s' that minimizes the error. Since each σ_s' corresponds to an equivalence class given by the order-1 similarity relation, this step allows us to select a class with minimal error. Then, we search for the best parameter point within this class by keeping σ_s' fixed and performing another 1D search to find the best σ_s. Figure 7.4-b shows an example starting from point S (the same initial guess as in Figure 7.4-a) where the first 1D search finds Q and the second returns R. Figure 7.5-c shows that this new solution matches the ground truth better under both the original (low-frequency) and the novel (high-frequency) lighting.

7.6 Application: forward rendering

Optically dense and forward-scattering materials are very common in the real-world [25, 31], but they are challenging to render. Our main application is to offer speedups to Monte Carlo rendering of these materials without modifying the core rendering algorithms. The basic idea is to find altered parameters with $\sigma_s^* < \sigma_s$ (namely, with $\alpha < 1$), so that fewer scattering events need to be handled.



(a) Reference (b) Original (c) Reparam.

Figure 7.5: Images rendered using the real solution (a) as well as solutions found by executing SGD on the original search space (b) and the reparameterized one (c). Visualizations of per-pixel relative error (using the color mapping in Figure 7.1) are included in (b, c). The images in the top row are used during the optimization process, and those in the bottom with a novel lighting are for validation. The solution found using the reparameterized space shown in (c) leads to better results in both configurations.

We introduce a practical algorithm (Algorithm 7.1) based on the theory introduced in Sections 7.3 and 7.4. To use this algorithm, the user can simply input the original scattering parameters and an extra number α which controls the balance between performance and accuracy, and the algorithm outputs a set of altered parameters corresponding to an optically thinner and less forward-scattering material. Rendering images using these altered parameters (without modifying the renderer) costs only a fraction of the computation required to render with the original ones. If the original parameters are spatially varying, the algorithm needs to be performed at each spatial location (or for each homogeneous region).

Next, we provide a detailed description of our technique: Section 7.6.1

presents an overview of the method (whose pipeline has been previewed in Figure 7.2); Section 7.6.2 describes the practical aspects of the user-specified parameter α; Section 7.6.3 introduces an "overfitting" problem and discusses how to avoid it. Detailed experimental evaluations of our technique are in Section 7.7.

7.6.1 Overview

Given the original scattering parameters σ_a, σ_s, $f(\cdot)$ and the user-specified parameter α, our method sets $\sigma_a^* = \sigma_a$ and $\sigma_s^* = \alpha\sigma_s$ (line 2 of Algorithm 7.1). Then, we determine the order N of the similarity relation to satisfy. Greater N normally provides better accuracy but may lead to unsatisfiable constraints. Thus, we use the IFEXISTS() routine implementing Theorem 1 (line 6) to find the highest order N of which the similarity relation is satisfiable (lines 4 to 10). In practice, we bound N with $N_0 = 5$ (line 5) because our experiments indicate that going beyond this order does not provide observable improvement for resulting quality.

Occasionally, higher-order relations (greater N) yield worse accuracy, and we call this problem "overfitting". This is another reason that we pick $N_0 = 5$ as relations beyond this order tend to overfit. Section 7.6.3 presents a simple method to reject the overfitting results (line 14 of Algorithm 7.1).

Algorithm 7.1 contains the pseudocode of the entire pipeline. Despite the sophistication of the underlying theory, the algorithm itself is very easy to implement.

Algorithm 7.1 Computing the altered scattering parameters.

function COMPUTEPARAMETERS($\sigma_a, \sigma_s, f(\cdot), \alpha$)
  $\sigma_a^* \leftarrow \sigma_a, \quad \sigma_s^* \leftarrow \alpha\sigma_s$
  Compute $f_1^*, \ldots, f_{N_0}^*$ using (7.23)
  $N \leftarrow 1$ ▷ Compute the order N
  while $N < N_0$ **do**
    if not IFEXISTS($1, f_1^*, \ldots, f_{N+1}^*$) **then** ▷ Theorem 1
      break
    end if
    $N \leftarrow N + 1$
  end while
  repeat ▷ Estimate $f^*(\cdot)$ numerically
    Solve the order-N version of (7.31) to obtain $f^*(\cdot)$
    $N \leftarrow N - 1$
  until $f^*(\cdot)$ is NOT overfitting ▷ Section 7.6.3
  return σ_a^*, σ_s^*, $f^*(\cdot)$
end function

7.6.2 User-specified parameter

We now describe the way in which the user-specified parameter α in Algorithm 7.1 affects performance and accuracy, and discuss how to pick the value of this parameter properly.

Balancing performance and accuracy. In practice, the parameter α controls the tradeoff between performance and accuracy: small α offers good performance but potentially poor accuracy; large α provides good accuracy but at the cost of slower performance.

When $\sigma_a^* \ll \sigma_s^*$ (namely the single-scattering albedo is high), it holds that $\alpha = \sigma_s^*/\sigma_s \approx (\sigma_s^* + \sigma_a^*)/(\sigma_s + \sigma_a) = \sigma_t^*/\sigma_t$. Therefore, along a unit distance within the altered material, the expected number of scattering events is roughly a factor α of that within the original material. For Monte Carlo methods where each scattering event is explicitly simulated, this usually means that α is linearly

related to the rendering time.[3] Experimental results demonstrating this effect are in Section 7.7.1.

Picking α. Although α can theoretically take any value in $(0, 1)$, not every value in this range leads to high-quality results. Assuming that the original material is forward-scattering ($f_1 > 0$), we found that the resulting accuracy decreases rapidly when $f_1^* < 0$. So we require $f_1^* \geq 0$ which implies

$$\alpha \geq 1 - f_1. \tag{7.32}$$

For objects with optically thin regions, relatively large α values are required to produce high-quality results. We observed that setting $\alpha = \max(0.3, 1 - f_1)$ worked quite well for all our experiments, even under conditions that are highly unforgiving to errors, such as back lighting. To fine-tune this parameter for greater speedups, a small number of test renderings can be performed as in Figure 7.7 but using a low resolution and a small number of random samples, so that it introduces little overhead and the resulting α can be reused for high-quality renderings or generating animated sequences. In this case, we suggest starting with a smaller α value such as $\max(0.1, 1 - f_1)$ and increasing α iteratively until the test renderings converge visually.

[3] With the presence of Russian roulette, the expected number of scattering events can be bounded, which may reduce the rendering time for large α and makes the relation between α and the rendering time more complicated. Because performing Russian roulette can introduce a great amount of noise when the single-scattering albedo σ_s/σ_t is close to 1, how to do it properly is non-trivial and beyond the scope of this work. Thus, we did not use this technique when creating our renderings.

7.6.3 Overfitting

Since overfitting does not happen very frequently and normally only causes subtle visual differences, we use a simple method to determine whether a solution $f^*(\cdot)$ is likely to overfit (based on its "support") and reject the overfitting solutions. Examples of overfitting are in Section 7.7.2 (Figure 7.8) and Section A.5 (Figure A.3).

Given a phase function $f(\cdot)$, we define its *(normalized) support* $\mathrm{nz}(f)$ to be a fraction between 0 and 1 that equals the portion of the domain where $f(\cdot)$ is greater than zero. Namely,

$$\mathrm{nz}(f) := \frac{|\{t : f(t) > 0\}|}{2}.$$

For a tabulated phase function $f^*(t)$ with k bins, $\mathrm{nz}(f^*) = k'/k$ where k' is the number of bins in which $f^*(t)$ is positive.

Intuitively, to approximate a translucent material with an altered one where scattering occurs less frequently, one needs to allow light to scatter into a wider range of directions. Our experiments indicate that given the moment constraints in (7.29), solution phase functions with larger supports provide better results. Therefore, we regularize $f^*(\cdot)$ by minimizing the 2-norm of the smoothness term, which favors support instead of sparsity, in Section 7.4.2.

On the other hand, for fixed α, the altered phase function needs to be more concentrated (namely, to have smaller support) for satisfying higher-order similarity relations. Occasionally, these relations become barely satisfiable, resulting in phase functions with very limited supports that tend to overfit.

To remedy this problem, we threshold the support of altered phase functions. Given the original phase function $f(\cdot)$, if the altered phase function $f^*(\cdot)$

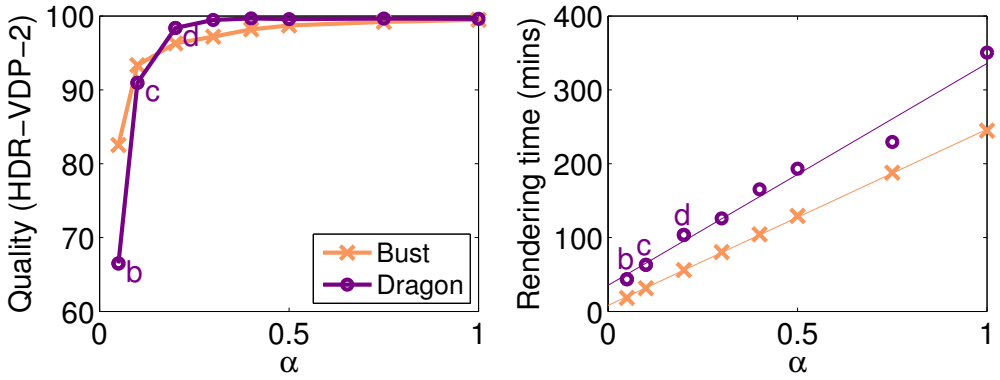
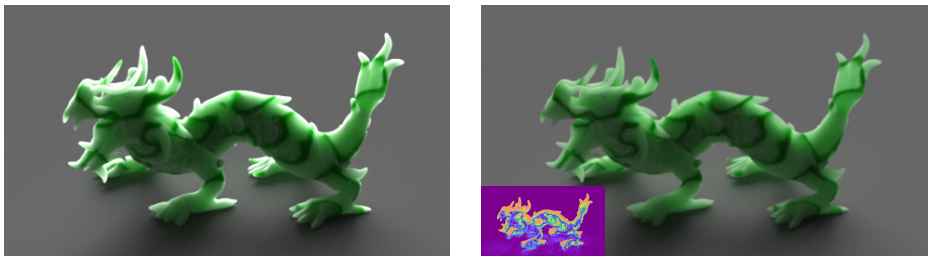


Figure 7.6: The rendering quality scores (evaluated using the HDR-VDP-2 metric) and the execution times when changing the value of α. Data points on the purple curves marked with 'b', 'c', and 'd' respectively correspond to renderings in Figure 7.7-bcd.

has $\mathrm{nz}(f^*) < \beta\ \mathrm{nz}(f)$ for some constant $\beta \in (0, 1]$, we reject it (line 14 of Algorithm 7.1). In our experiments, we used $\beta = 0.65$.

7.7 Experimental results

In this section, we first show how the choice of α in Algorithm 7.1 balances performance and accuracy (Section 7.7.1). Next, Section 7.7.2 demonstrates that considering similarity relations beyond order-1 can provide much better accuracy for complex phase functions. Then, we show that higher-order analysis is required to capture perceptually significant cues proposed by Gkioulekas et al. [29] (Section 7.7.3). Finally, Section 7.7.4 exhibits rendered results for a variety of translucent media. All our renderings are created using the Mitsuba physically-based renderer [42].



(a) Reference
350 minutes

(b) $\alpha = 0.05$
44 minutes, Score: 66.49

(c) $\alpha = 0.1$
63 minutes, Score: 90.94

(d) $\alpha = 0.2$
103 minutes, Score: 98.39

Figure 7.7: **Renderings of a heterogeneous dragon:** (a) ground truth; (b, c, d) renderings using the altered parameters generated using Algorithm 7.1 with different α values. As in Figure 7.1, the relative error visualizations are included.

7.7.1 Performance versus accuracy

We now show how the choice of α affects the performance of Monte Carlo path tracing and the resulting accuracy through experiments.

We created two scenes each of which contains a translucent object lit under high-frequency environment lighting [14]. Both objects have HG phase functions with $g = 0.95$, and the sizes of these objects are several hundreds times the mean free path. Then, we generated 7 sets of altered parameters using Algorithm 7.1 with α ranging from 0.05 to 0.75.

Figure 7.6 shows the rendering quality (evaluated using the HDR-VDP-2 perceptual metric [66] where a higher score means better quality) and the execution time (using standard volume path tracing) as functions of α. Note that, in prac-

tice, the speedup does not exactly equal $1/\alpha$ as the rendering time is affected by many factors varying among different α values, such as cache performance (which is higher for greater α since the rendering algorithm tends to access data with better locality).

Figure 7.7 shows some of the rendered images corresponding to the purple curves in Figure 7.6. More renderings are presented in Appendix A (Section A.4). We can see that when $\alpha = 0.05$, while a 8.0X speedup can be achieved, the resulting accuracy is unsatisfactory (Figure 7.7-b). On the other hand, with $\alpha = 0.1$ or 0.2 (which respectively offer speedups of 5.5X and 3.4X), significantly better accuracy can be obtained (Figure 7.7-cd).

7.7.2 Higher-order similarity relations

Although first-order approximations work adequately for simple phase functions (such as single-lobe HG), they do not have sufficient representative power to capture higher moments of $f(\cdot)$.

We took a phase function proposed by Gkioulekas et al. [29] which is a linear combination of two distributions:

$$f(\cos\theta) = 0.9\ \mathrm{HG}(0.95, \cos\theta) + 0.1\ \mathrm{vMF}(-75, \cos\theta) \tag{7.33}$$

where $\mathrm{HG}(g, \cdot)$ and $\mathrm{vMF}(\kappa, \cdot)$ denote the HG function with parameter g and the von Mises-Fisher distribution with parameter κ, respectively. Then, we solved the quadratic programming problem in (7.31) to construct three altered versions of (7.33) adhering to the order-1, order-4, and order-5 similarity relations, respectively. The first Legendre moments of these altered phase functions are all zero. Figure 7.8-a plots (7.33) and its altered versions.

Figure 7.8-bcde contains a homogeneous dragon rendered using these phase functions. We can see that the accuracy offered by the order-1 version (Figure 7.8-c) is not ideal as it is not able to capture higher-order features of the original phase function. The renderings produced using the order-4 version (Figure 7.8-d), on the other hand, match the ground truth very well. The order-5 version (the orange curve in Figure 7.8-a) has a fairly low support and does not generalize to different lighting conditions as well as the order-4 one (Figure 7.8-e), although the visual difference is subtle. Thus, this solution is overfitting and will be rejected by Algorithm 7.1 (which instead returns the order-4 version). Please refer to Appendix A for more examples on overfitting.

7.7.3 Spanning the 2D perception space

Next, we evaluate our method on a family of phase functions [29] spanning a 2D perception space. The two axes of this space capture the optical density (vertical) and the level of "glass-like" appearance (horizontal).

We picked 40 representatives from this family and rendered an image for each of them (with the absorption and scattering coefficients fixed). Figure 7.9-c shows two of these renderings. Then, we applied classical multidimensional scaling (MDS) to create a 2D embedding of the renderings where a linear transform T assigns each image a 2D coordinate. The blue squares in Figure 7.9-ab illustrate this 2D embedding.

Next, for each phase function, we computed an altered set of parameters satisfying similarity relations up to order-5 (using Algorithm 7.1) and rendered an image accordingly. Two of these rendered images are in Figure 7.9-d. Then, we projected those images into the previously created 2D space (using the same op-

erator T). Figure 7.9-a shows that the projections can well maintain the structure of the original embedding. The small offsets between corresponding points in the two embeddings cause little visible difference, and some of them are caused by the Monte Carlo noise.

On the other hand, if we use a configuration which satisfies only the order-1 relation, the resulting renderings are missing important visual cues (as illustrated in Figure 7.9-e), causing their projections to collapse to a 1D line (Figure 7.9-b).

Figure 7.9 demonstrates that higher-order analysis is crucial to accurately capture perceptually significant visual cues. Section A.6 contains all the images used to create the three embeddings in Figure 7.9.

7.7.4 Rendered results

Next, we demonstrate that the altered scattering parameters generated by Algorithm 7.1 produce appearances that accurately match the ground truth under a variety of scene configurations.

Figures 7.1 and 7.10 exhibit rendered images created using volume path tracing. Each reference rendering contains an object with a spatially invariant phase function. Consequently, only one altered version needs to be computed for each result, which takes less than a second using our MATLAB implementation of Algorithm 7.1.

Figure 7.1 contains a Corinthian capital made of a homogeneous material with a complicated phase function (the one marked as 'A' in Figure 7.9). It also has a refractive interface modeled using the microfacet model [98]. Altered

parameters generated by our method offer a 3.7X speedup, and the resulting images match the ground truth very well.

In the first row of Figure 7.10, we show a sculpture made of a highly heterogeneous material where the white regions are about 5 times as dense as the green ones, which is easily visible under back lighting as shown in (a). The sculpture has the complex phase function described in (7.33) and a rough dielectric interface. Our method offers a 2.7X speedup, and the results match the ground truth very well in both the thick and the thin regions.

The second row of Figure 7.10 contains renderings of a highly scattering smoke volume which has a HG phase function with $g = 0.95$. Parameters provided by our method, which satisfy the order-5 similarity relation, lead to a 3.5X speedup while maintaining good accuracy even at highly thin regions.

In the third row, we show rendered images of a homogeneous bust made of a material with a complicated phase function and a rough dielectric interface. Parameters provided by our method speeds up the rendering process by 3.4X, and the resulting images match the reference quite well.

7.8 Conclusion

In this chapter, we present a complete exposition of similarity theory, providing fundamental insights into the structure of the RTE's parameter space. Furthermore, we develop a novel approach to solve for the altered parameters satisfying the similarity relation of any given order. Since the altered phase function is not fully specified by the relations, we present the sufficient and necessary conditions for its existence and introduce a numerical algorithm to find one (when it

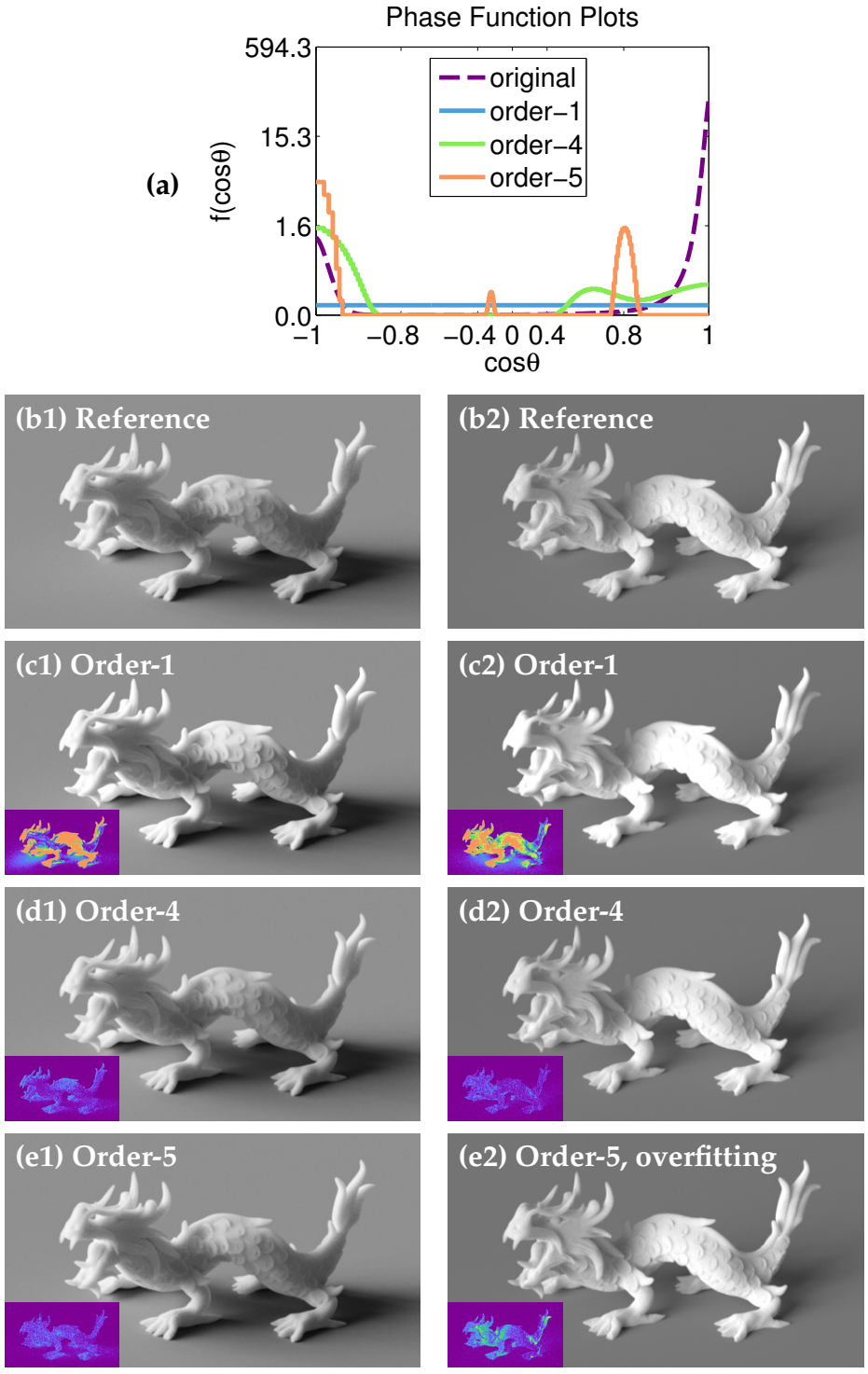
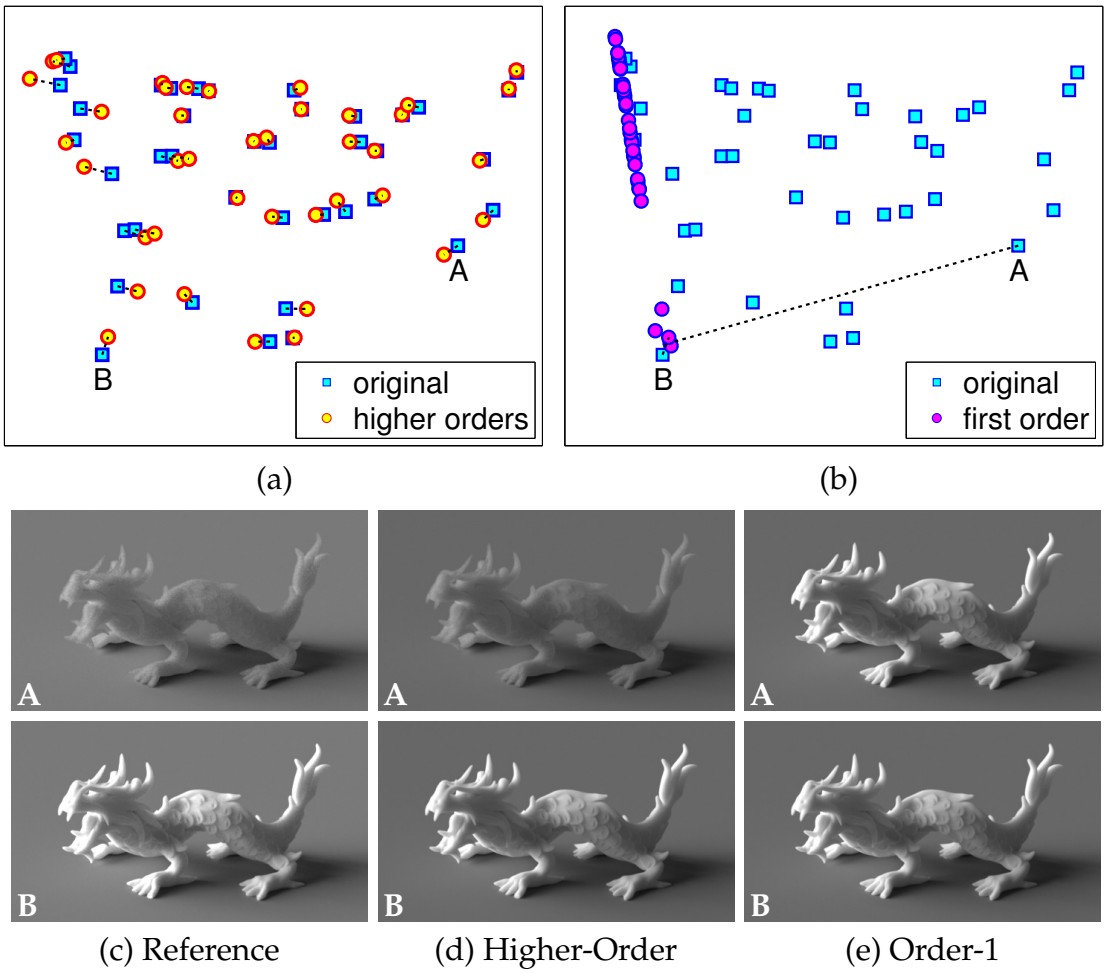


Figure 7.8: A complicated phase function and its three altered versions respectively satisfying the order-1, order-4, and order-5 similarity relations are plotted in (a). Renderings of a homogeneous dragon (using the plotted phase functions) under side lighting (left) and front lighting (right) are in (b, c, d, e). The order-1 version yields poor accuracy; the order-5 version works adequately but not as well as the order-4 one under both lighting conditions.



(a) (b)

(c) Reference (d) Higher-Order (e) Order-1

Figure 7.9: **2D embeddings:** (a) altered parameters satisfying up to order-5 similarity relations can well maintain the structure of the original embedding; (b) satisfying only the order-1 relation causes the projections to collapse to a 1D line. The dashed lines in (a, b) connect the projections of images rendered with the original and the altered parameters. The remaining columns show renderings of two phase functions (marked with A and B) which have similar first moments: (c) reference renderings, (d, e) images rendered using altered parameters adhering to higher-order relations and the order-1 relation, respectively. As demonstrated in (e), first-order approximations do not have sufficient representative power to distinguish these phase functions (such as A and B), causing them to be mapped to similar locations in (b).

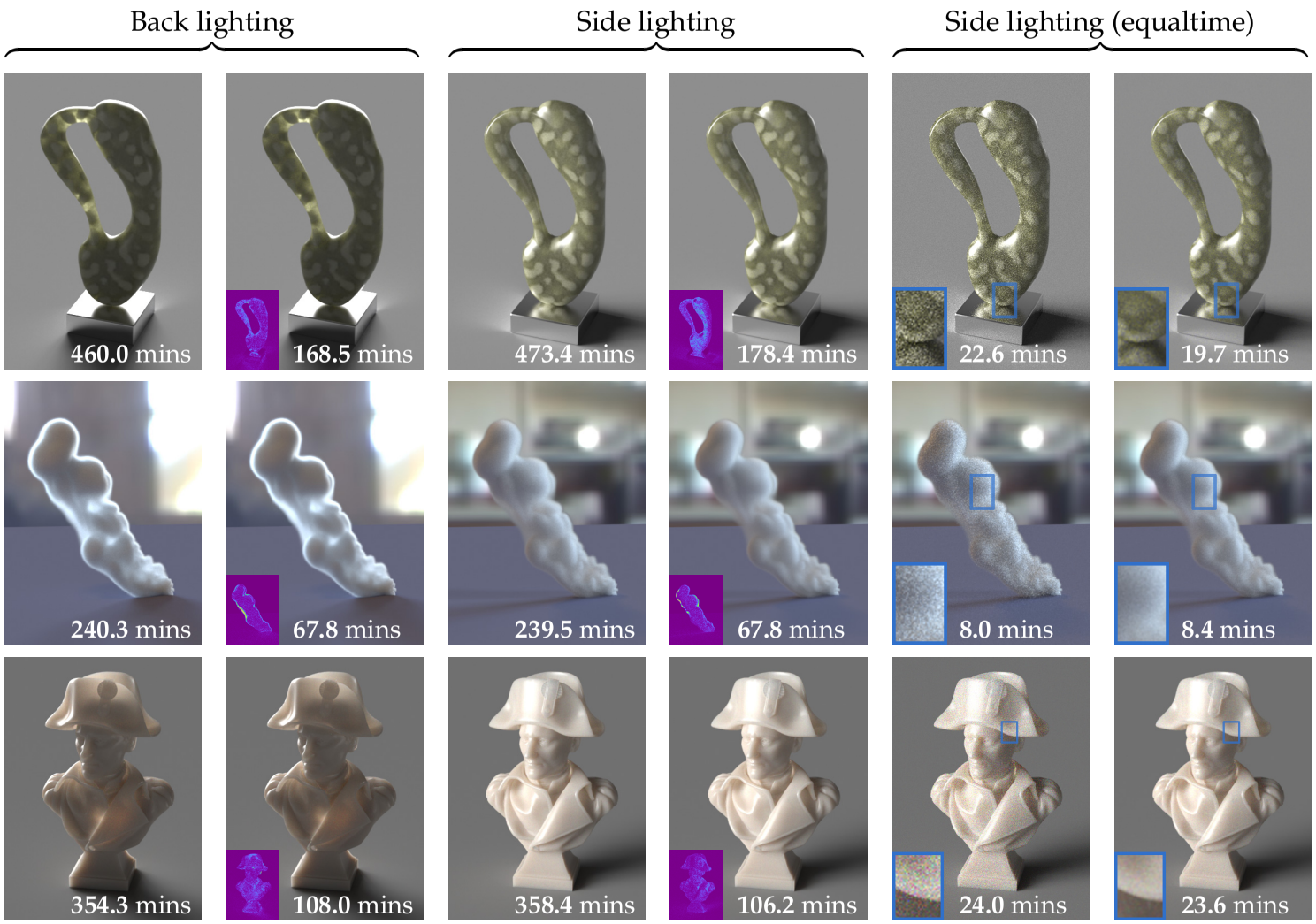


Figure 7.10: **Path-traced renderings** with various scene configurations. Columns (a, b) contain images rendered using the original and the altered parameters with the same number of sample paths per pixel. Column (c) uses the same scene configuration as (b) and shows images rendered in similar time with both parameters (see the insets to assess noise). The relative error maps (using the color scheme in Figure 7.1) are included in (a2, b2).

exists) that can produce high-quality renderings.

We use two applications, forward and inverse rendering of translucent media, to demonstrate the practical utility of our theory. For inverse rendering, we present a reparameterized search space to overcome the challenges caused by the presence of equivalence classes. For forward rendering, our main application, we develop an approach (Algorithm 7.1) to offer speedups to Monte Carlo rendering of optically dense and forward-scattering media without having to modify the core rendering algorithms. Despite the sophistication of the theory, our method is very easy to implement and introduces negligible overhead to the full rendering pipeline.

There are many areas of future work. For inverse rendering, we would like to study how similarity theory can improve solving material appearance acquisition problems with high-dimensional search spaces on real data. For forward rendering, we plan to explore heuristics for choosing the value of α adaptively, which is a generalization of the hybrid framework, so that greater speedups can be obtained. In addition, the phase functions used in our results have no spatial variation. For spatially varying phase functions, computing many altered versions can be costly, and interpolating them without violating the similarity relation constraints is a non-trivial problem. Thus, we intend to develop approaches to efficiently compute and properly interpolate spatially varying altered phase functions. Finally, for the theory, we would like to investigate how the generalized similarity relations, such as (7.19), could benefit computer graphics applications.

CHAPTER 8

CONCLUSION

Fabrics are one of the most common materials in our everyday lives. Modeling and rendering them with high fidelity can benefit not only computer graphics but also other industrial applications including textile design, retail and entertainment.

This thesis has brought a new level of realism to computer generated imagery of fabrics. Our contributions include:

- A fundamentally new approach to build volumetric appearance models for fabrics with micron-resolution (Chapter 4). We have demonstrated that by acquiring and explicitly modeling fiber-level details, fabric renderings with unprecedented visual quality can be obtained.
- A structure-aware synthesis algorithm that creates high fidelity models with user-specified weave patterns (Chapter 5). This approach allows previewing general textile designs before physically constructing them.
- A precomputation based algorithm that accelerates the rendering of our micron-resolution fabric models by an order of magnitude (Chapter 6), significantly improving their practical usefulness.

In addition, we have made a theoretical contribution by introducing (high-order) similarity theory to computer graphics (Chapter 7) which provides significant insights on the structure of the parameter space of translucent media. We have also developed a practical algorithm to utilize this theory in its most general form and benefit both forward and inverse rendering of translucent media.

8.1 Future research directions

Reproducing and predicting the appearance of complex materials has long been a grand challenge. In this dissertation, we have taken one step forward by introducing new approaches in predictive appearance modeling and efficient rendering of fabrics as well as in general light transport theory. We believe that these techniques can inspire future research that has the potential to significantly benefit applications in not only graphics but also many other fields such as industrial design and architecture.

Appearance Modeling and Fabrication

High-quality models capturing how light interacts with a material are essential to reproducing and predicting its appearance. Unfortunately, many complex materials are not handled well enough with existing methods. Thus, it is necessary to explore new means to model light transport in those materials.

Appearance fabrication. Recently, bringing objects from the virtual world to reality using manufacturing techniques such as 3D printing has been an active research area. Thanks to the advances of those techniques, we are allowed to control not only the geometry but also the appearance of a fabricated object. However, building physical objects with user-specified appearance while respecting various constraints from the manufacturing process remains challenging. This problem may be tackled by introducing appearance models with parameters meaningful to the fabrication process and deriving new algorithms to efficiently solve for the parameter values.

Appearance acquisition. We cannot model a material's appearance without knowing precisely how it appears. Namely, we need to acquire the appearance under calibrated configurations. This, however, is not easy: brute-force approaches often require expensive hardware such as gonioreflectometers or specialized cameras, and such acquisition processes can take many hours, if not days. Therefore, new acquisition systems that are both cost-efficient and offer great accuracy need to be designed. Such systems can be very useful for numerous applications in computer graphics and vision.

Appearance models. Based on appearance measurements, we can derive data-driven models. These models need to be compact and easy-to-evaluate, so that they can be efficiently used by state-of-the-art rendering algorithms. Since the measurements usually include large amounts of data (due to the development of digital photography), how to compress the measured appearance data needs to be investigated. Alternatively, models that are not limited to one kind of material but generalize to a large variety of them are of great value. Comparing to the data-driven ones, these models normally build upon physical principles of light-medium interaction.

Automated model creation. Having the appearance models is of limited practical value unless we have algorithms to instantiate them. Namely, for the new appearance models that we would like to develop, it is important to also build efficient algorithms to create model data while requiring minimal amount of user effort. One key problem we will need to solve is "inverse rendering": finding proper model parameters which yield desired appearance. This boils down to solving challenging optimization problems in which certain quantities (such

as the radiance) can only be estimated using Monte Carlo simulation.

Realistic Rendering

Efficient and easy-to-control algorithms are essential for utilizing appearance models. Unfortunately, extremely challenging problems remain. Furthermore, as new appearance models are developed, new rendering problems emerge.

Efficient rendering algorithms. Photo-realistic rendering has long been computationally intensive: correctly simulating light transport in large scenes requires great amount of computation. This is even more true now due to the ever-increasing complexity of both object geometries (3D models) and materials. This problem could be addressed in two complementary manners. First, for materials with characteristic structures, specialized approaches can be developed to exploit them and improve performance. Second, general methods approximating the radiative transfer process can be derived.

Interactive material editing. Getting interactive feedback is critical to many industrial design applications. Unfortunately, realistic rendering algorithms in general do not offer this level of performance. Thus, algorithms handling material changes interactively are worth investigating. One possibility is to precompute and store light transport information while symbolically carrying the material related parameters on the fly. Naïve implementations of this idea, however, normally yield unacceptable amount of precomputation or storage. Thus, we will need to answer the questions on how to represent and store the precomputed information, and how to use them in the run-time for efficient simulation of light transport.

APPENDIX A

APPENDIX FOR CHAPTER 7

A.1 Derivation of the diffusion equation

In this section, we present an alternative derivation of the diffusion equation (DE) based on the same assumption taken by the order-1 similarity relation: the radiance field L is *linearly anisotropic*, namely

$$a_{mn} = 0 \quad \text{for all} \quad n > 1. \tag{A.1}$$

We first rewrite the RTE (7.1) in a series of integrated forms (Section A.1.1). Then, we present in Section A.1.2 the order-0 and order-1 versions of the integrated RTE and simplify them based on the assumption (A.1). Finally, we combine the results from Section A.1.2 to obtain the DE (Section A.1.3).

A similar derivation has been proposed by Wyman et al. [106] to obtain the generalized similarity relations described in Section A.2.

A.1.1 Integrated RTE

Let

$$\omega_{i_1 i_2 \ldots i_n} := \prod_{j=1}^{n} \omega_{i_j}$$

where ω_{i_j} denotes the i_j-th component of $\boldsymbol{\omega}$, which is a 3-vector (so $i_j \in \{1, 2, 3\}$ for all j). In addition, we define the integrated forms of the three terms (includ-

ing the source term Q) appearing in the RHS of the RTE (7.1):

$$\phi_{i_1 i_2 \ldots i_n} := \int_{\mathbb{S}^2} \omega_{i_1 i_2 \ldots i_n} L(\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega}, \tag{A.2}$$

$$\chi_{i_1 i_2 \ldots i_n} := \int_{\mathbb{S}^2} \int_{\mathbb{S}^2} \omega_{i_1 i_2 \ldots i_n} f(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) L(\boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}' \, \mathrm{d}\boldsymbol{\omega}, \tag{A.3}$$

$$Q_{i_1 i_2 \ldots i_n} := \int_{\mathbb{S}^2} \omega_{i_1 i_2 \ldots i_n} Q(\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega}. \tag{A.4}$$

An integrated version of the RTE (7.1) can then be obtained by multiplying $\omega_{i_1 i_2 \ldots i_n}$ and integrating over $\mathbb{S}^2$ on both sides, yielding

$$\sum_{j=1}^{3} \partial_j \phi_{j i_1 \ldots i_n} = -\sigma_t \phi_{i_1 \ldots i_n} + \sigma_s \chi_{i_1 \ldots i_n} + Q_{i_1 \ldots i_n} \tag{A.5}$$

where ∂_i denotes $\frac{\partial}{\partial \omega_i}$.

A.1.2 Order-0 and order-1 versions

We now write down the order-0 and order-1 versions of the integrated RTE (A.5) and simplify them based on (A.1).

Order-0. The order-0 version of (A.5) is

$$\sum_{j=1}^{3} \partial_j \phi_j = -\sigma_t \phi + \sigma_s \chi + Q_{(0)} \tag{A.6}$$

where $Q_{(0)} := \int_{\mathbb{S}^2} Q(\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega}$. Since

$$\chi = \int_{\mathbb{S}^2} L(\boldsymbol{\omega}') \int_{\mathbb{S}^2} f(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega} \, \mathrm{d}\boldsymbol{\omega}' = \int_{\mathbb{S}^2} L(\boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}' = \phi,$$

(A.6) simplifies to

$$\sum_{j=1}^{3} \partial_j \phi_j = -\sigma_a \phi + Q_{(0)}. \tag{A.7}$$

Order-1. The order-1 version of (A.5) is

$$\sum_{j=1}^{3} \partial_j \phi_{ji} = -\sigma_t \phi_i + \sigma_s \chi_i + Q_i. \tag{A.8}$$

Due to (A.1), it holds that

$$\begin{aligned}
\chi_i &= \int_{\mathbb{S}^2} \int_{\mathbb{S}^2} \omega_i f(\boldsymbol{\omega}' \cdot \boldsymbol{\omega}) L(\boldsymbol{\omega}') \,\mathrm{d}\boldsymbol{\omega}' \,\mathrm{d}\boldsymbol{\omega} \\
&= \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \sum_{i=0}^{1} \sum_{j=-i}^{i} f_n a_{ji} \\
&\quad \left(\int_{\mathbb{S}^2} \omega_i Y_n^m(\boldsymbol{\omega}) \underbrace{\int_{\mathbb{S}^2} \bar{Y}_n^m(\boldsymbol{\omega}') Y_i^j(\boldsymbol{\omega}') \,\mathrm{d}\boldsymbol{\omega}'}_{=\delta_{ni}\ \delta_{mj}} \,\mathrm{d}\boldsymbol{\omega} \right) \\
&= \sum_{n=0}^{1} \sum_{m=-n}^{n} f_n a_{mn} \int_{\mathbb{S}^2} \omega_i Y_n^m(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} \\
&= f_1 \sum_{m=-1}^{1} a_{m1} \int_{\mathbb{S}^2} \omega_i Y_1^m(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega}
\end{aligned} \tag{A.9}$$

where the last equality follows the fact that $\int_{\mathbb{S}^2} \omega_i Y_0^0(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} = 0$ for all $i \in \{1, 2, 3\}$. Similarly, we have

$$\phi_i = \int_{\mathbb{S}^2} \omega_i L(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} = \sum_{m=-1}^{1} a_{m1} \int_{\mathbb{S}^2} \omega_i Y_1^m(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega}. \tag{A.10}$$

From (A.9) and (A.10), we know that

$$\chi_i = f_1 \phi_i. \tag{A.11}$$

It is easy to verify that for all i_1 and i_2, $\int_{\mathbb{S}^2} \omega_{i_1 i_2} Y_0^0(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} = \frac{2\sqrt{\pi}}{3} \delta_{i_1 i_2}$ and $\int_{\mathbb{S}^2} \omega_{i_1 i_2} Y_1^m(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} = 0$. Thus,

$$\phi_{i_1 i_2} = \int_{\mathbb{S}^2} \omega_{i_1 i_2} L(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} = a_{00} \frac{2\sqrt{\pi}}{3} \delta_{i_1 i_2}.$$

Note that $a_{00} = \int_{\mathbb{S}^2} Y_0^0(\boldsymbol{\omega}) L(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} = \frac{\phi}{2\sqrt{\pi}}$, we have

$$\phi_{i_1 i_2} = \frac{\phi}{3} \delta_{i_1 i_2}. \tag{A.12}$$

Substituting (A.11) and (A.12) into (A.8) leads to

$$\frac{1}{3}\partial_i\phi = -\sigma_{tr,1}\phi_i + Q_i \tag{A.13}$$

where $\sigma_{tr,1} = \sigma_t - f_1\sigma_s$ follows the definition in (7.11).

A.1.3 Diffusion equation

In this subsection, we show that the DE can be obtained easily from the integrated RTE (A.5) in its simplified order-0 (A.7) and order-1 (A.13) forms.

Let

$$\mathbf{E} := \int_{\mathbb{S}^2} \boldsymbol{\omega} L(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix}, \quad \mathbf{Q}_{(1)} := \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \end{pmatrix},$$

and $\kappa := (3\sigma_{tr,1})^{-1}$. Then (A.7) and (A.13) can be rewritten as

$$\nabla \cdot \mathbf{E} = -\sigma_a\phi + Q_{(0)}, \tag{A.14}$$

$$\mathbf{E} = 3\kappa\,\mathbf{Q}_{(1)} - \kappa\nabla\phi \tag{A.15}$$

where $\nabla\cdot$ is the divergence operator and ∇ is the gradient operator. Substituting (A.15) into (A.14) yields

$$\nabla \cdot \big(3\kappa\,\mathbf{Q}_{(1)} - \kappa\nabla\phi\big) = 3\kappa\,\nabla\cdot\mathbf{Q}_{(1)} - \nabla\cdot(\kappa\nabla\phi) = -\sigma_a\phi + Q_{(0)}.$$

Namely,

$$-\nabla\cdot(\kappa\nabla\phi) + \sigma_a\phi = Q_{(0)} - 3\kappa\,\nabla\cdot\mathbf{Q}_{(1)}, \tag{A.16}$$

and (A.16) is known as the *diffusion equation*.

A.2 Generalized order-1 similarity relation

The similarity relations (7.18) requires $\sigma_a^* = \sigma_a$ to ensure that the altered RTE (7.2) has the same solution radiance field L as the original (2.1). However, if we relax this equal-radiance constraint and only ask for the same *fluence* $\phi = \int_{\mathbb{S}^2} L(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}$, generalized versions of the similarity relations can be derived [106]. Unlike (7.18), unfortunately, these relations do not easily generalize to higher orders.

Assuming (A.1) holds, namely the radiance field is linearly anisotropic, (A.7) and (A.13) can be used to derive the generalized similarity relation of order-1 as follows. Applying ∂_i and summing over 1 to 3 on both sides of (A.13) and substituting (A.7) into the resulting equation yields

$$\sum_{i=1}^{3} \partial_i^2 \phi = \sigma_a \sigma_{tr,1} \phi - \sigma_{tr,1} Q_{(0)} + \sum_{i=1}^{3} \partial_i Q_i.$$

For participating media with no internal source, all moments of Q vanish, giving $\sigma_a \sigma_{tr,1}\phi - \sum_i \partial_i^2 \phi = 0$. Similarly, it holds that $\sigma_a^* \sigma_{tr,1}^* \phi - \sum_i \partial_i^2 \phi = 0$. Equating these two equations yields

$$\sigma_a \sigma_{tr,1} = \sigma_a^* \sigma_{tr,1}^*, \tag{A.17}$$

which is the order-1 generalized similarity relation. In addition, it has been shown in [106] that the solution radiance field L of the original RTE and the solution L^* of the altered one are related as follows:

$$L(\boldsymbol{\omega}) = \frac{\sigma_a}{\sigma_a^*} L^*(\boldsymbol{\omega}) + \left(1 - \frac{\sigma_a}{\sigma_a^*}\right) \frac{\phi}{4\pi}. \tag{A.18}$$

Note that when $\sigma_a^* = \sigma_a$, (A.18) collapses to $L(\boldsymbol{\omega}) = L^*(\boldsymbol{\omega})$ and (A.17) becomes the order-1 similarity relation, namely (7.18) with $N = 1$.

A.3 Phase function moments

Given a phase function $f(\cdot)$, its *monomial moments* and *Legendre moments* are defined as follows.

Monomial Moments. Given $f(\cdot)$, its n-th monomial moment is

$$\gamma_n := \int_{-1}^{1} f(t)\, t^n \,\mathrm{d}t. \tag{A.19}$$

Legendre Moments. Given $f(\cdot)$, its n-th Legendre moment is

$$f_n := 2\pi \int_{-1}^{1} f(t) P_n(t) \,\mathrm{d}t \tag{A.20}$$

where P_n denotes the degree-n Legendre polynomial and 2π is a normalization term. Since $f(\cdot)$ is a phase function, it holds that

$$f_0 = 2\pi \int_{-1}^{1} f(t) \,\mathrm{d}t = \int_{\mathbb{S}^2} f(\cos\theta) \,\mathrm{d}\boldsymbol{\omega} \equiv 1 \tag{A.21}$$

and

$$f_1 = 2\pi \int_{-1}^{1} t f(t) \,\mathrm{d}t = \int_{\mathbb{S}^2} \cos\theta f(\cos\theta) \,\mathrm{d}\boldsymbol{\omega} \tag{A.22}$$

where θ is the angle between $\boldsymbol{\omega}$ and the incident direction. f_1 is usually referred to as the *average cosine* of $f(\cdot)$. For a Henyey-Greenstein (HG) phase function with parameter g, its n-th Legendre moment has been shown to equal g^n (see Theorem 7.1 in [94]).

In fact, monomial moments and Legendre moments are closely related. Precisely, for any $N \geq 0$, the first $(N+1)$ Legendre moments and the first $(N+1)$ monomial moments *uniquely determine* each other. In other words, any phase function $f(\cdot)$ with $f_0, \ldots, f_N$ fixed will have the same $\gamma_0, \ldots, \gamma_N$, and vice versa.

Proof. Let $P_n(t) = \sum_{i=0}^{n} p_{n,i}\, t^i$. According to (A.19) and (A.20), we know that for any $n \geq 0$,

$$f_n = 2\pi \sum_{i=0}^{n} \left(p_{n,i} \underbrace{\int_{-1}^{1} f(t)\, t^i \, \mathrm{d}t}_{=\gamma_i} \right). \tag{A.23}$$

Let $\boldsymbol{\gamma} := (\gamma_0\ \gamma_1\ \ldots\ \gamma_N)^T$ and $\mathbf{f} := (f_0\ f_1\ \ldots\ f_N)^T$, equation (A.23) can be rewritten in vector form as

$$\mathbf{f} = 2\pi\, \hat{\mathbf{P}}\, \boldsymbol{\gamma}. \tag{A.24}$$

where $\hat{\mathbf{P}}$ is an $(N+1) \times (N+1)$ matrix with $\hat{P}(i+1, j+1) = p_{i,j}$ for $0 \leq i, j \leq N$. It is easy to verify that $\hat{\mathbf{P}}$ is lower-triangular and non-singular. Thus, $\boldsymbol{\gamma}$ and $\mathbf{f}$ uniquely determine each other. Given $\mathbf{f}$, $\boldsymbol{\gamma}$ can be computed by solving the linear system in (A.24). ■

A.4 Results: performance versus accuracy

Figures A.1 and A.2 contain rendered images we used to study how α, the user-specified parameter, controls the balance between performance and accuracy.

The renderings in Figure A.1 correspond to the orange curve in Figure 7.6 in Chapter 7. The environment lighting used in the following rendering is "Kitchen" from [14]. Note that the error in the results with high α values are mostly from Monte Carlo noise.

The renderings in Figure A.2 correspond to the purple curve in Figure 7.6. We used the environment lighting "Ennis" from [14].

A.5 Results: overfitting

In our experiments, overfitting does not happen very often and normally causes only subtle differences. Figure A.3 illustrates two examples where altered parameters adhering to the order-3 similarity relation overfit. Our method (Algorithm 7.1) successfully rejects these solutions (and returns the order-2 versions) since their coverages are too small (which is demonstrated by the phase functions plots).

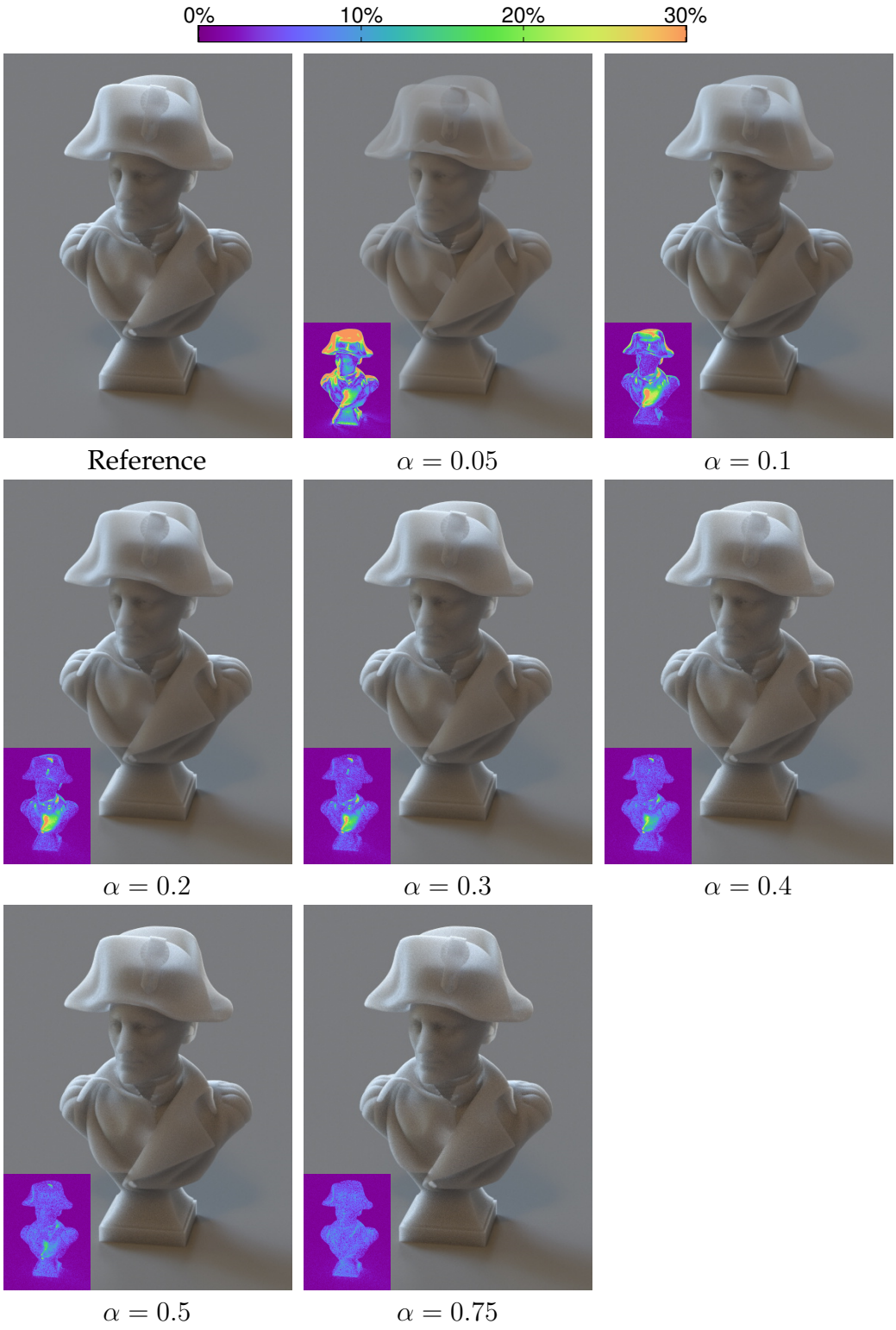


Figure A.1: **Rendered images** used to create the orange curve in Figure 7.6. The relative error maps (using the color mapping shown at the top) are included. Note that the error decreases with increasing α.

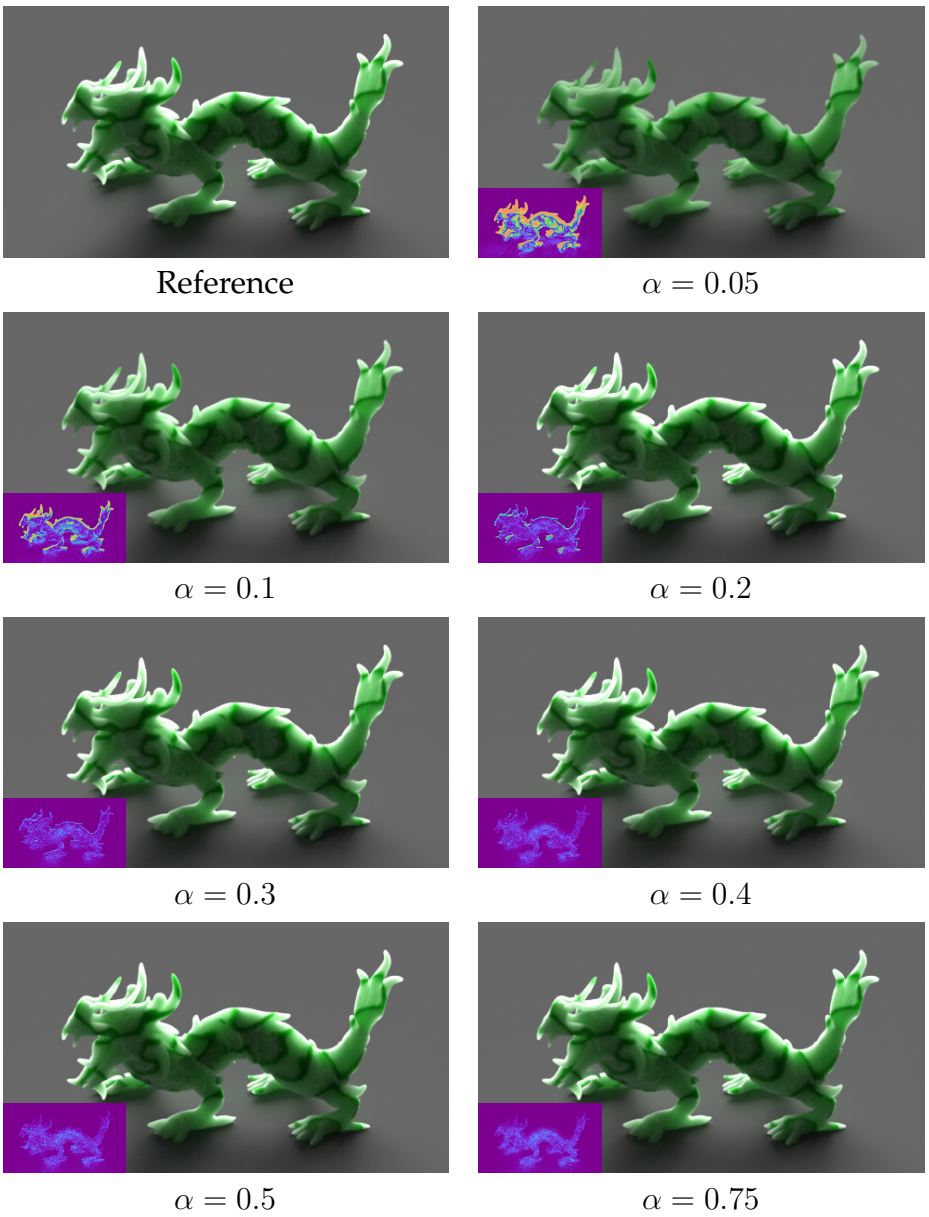


Figure A.2: **Rendered images** corresponding to the purple curve in Figure 7.6. The relative error visualizations use the same color mapping as Figure A.1.

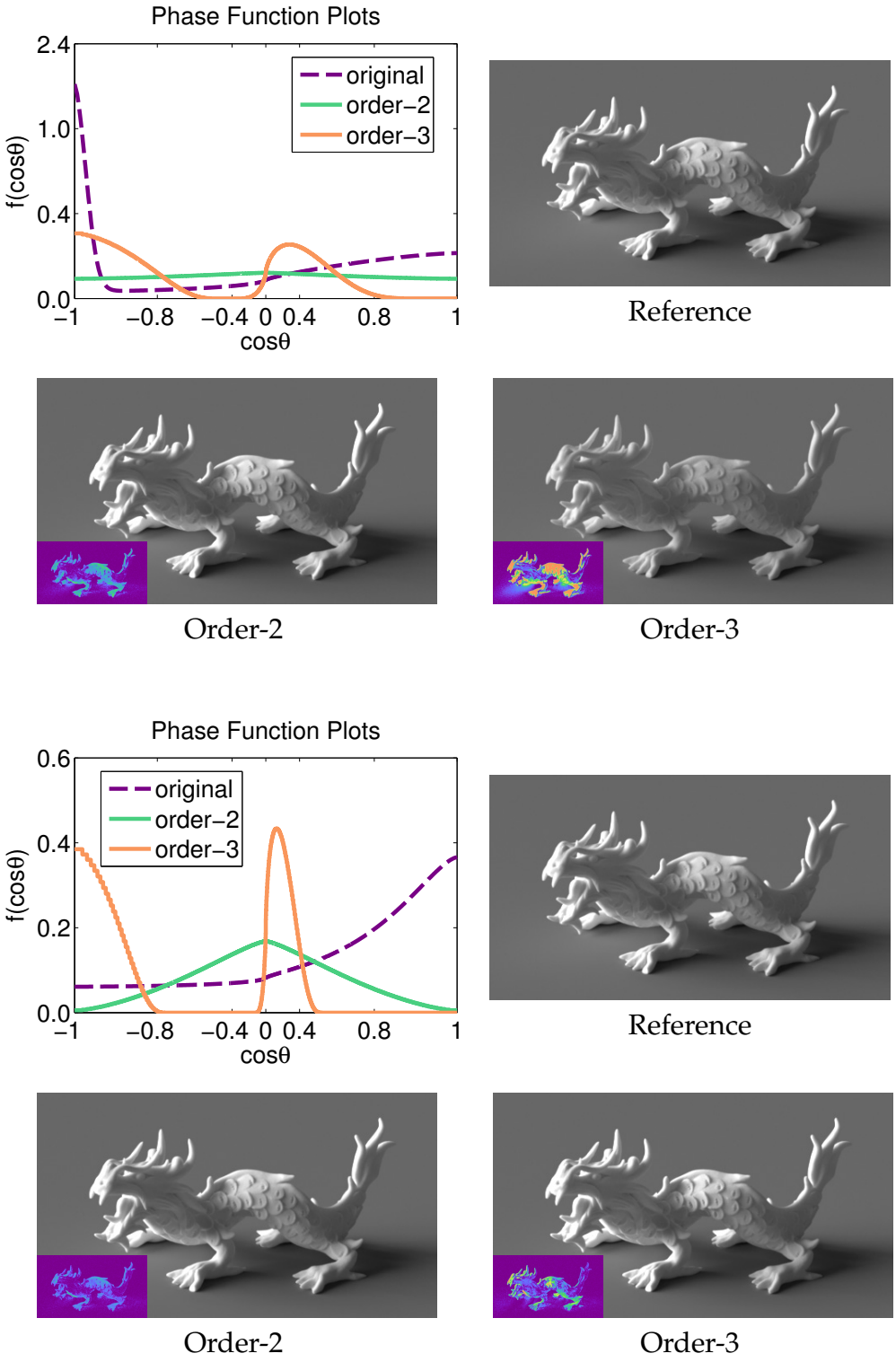


Figure A.3: **Two examples of overfitting.** In both examples, the altered parameters satisfying the order-3 similarity relation overfit.

A.6 Results: spanning the perceptual space

This section includes the images we used to create the embeddings in Figure 7.9. The two examples in Figure 7.9-cde correspond to phase functions 18 and 35.

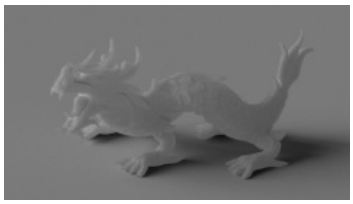
A.6.1 Rendered images

This section illustrates 40 rows of images, and each row corresponds to a phase function that we picked from [29]. In each row, the left image is rendered with the original parameters and used for creating the reference embedding (the blue points in Figure 7.9-ab). The middle image is generated with the altered parameters provided by Algorithm 7.1 in Chapter 7. We use it for the higher-order embedding (the yellow points in Figure 7.9-a). The right image is created with the altered parameters satisfying only the order-1 similarity relation. The corresponding embedding (which collapses to a 1D line) is in Figure 7.9-b. The middle and right images share the same altered scattering coefficient σ_s^* and are rendered in similar time.

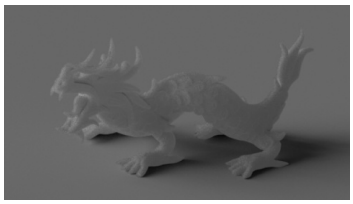
1: Reference

1: Order-3

1: Order-1



2: Reference



2: Order-5

2: Order-1

3: Reference

3: Order-5

3: Order-1

4: Reference

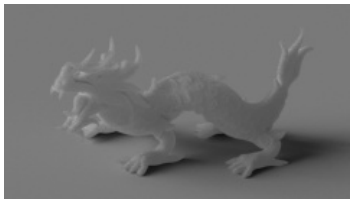
4: Order-5

4: Order-1

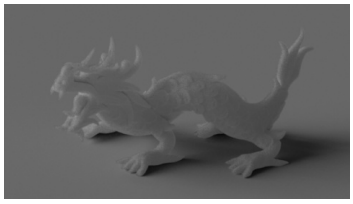
5: Reference

5: Order-5

5: Order-1



6: Reference



6: Order-4

6: Order-1

7: Reference

7: Order-3

7: Order-1

8: Reference

8: Order-2

8: Order-1

9: Reference

9: Order-3

9: Order-1

10: Reference

10: Order-4

10: Order-1

11: Reference

11: Order-3

11: Order-1

12: Reference

12: Order-3

12: Order-1

13: Reference

13: Order-2

13: Order-1

14: Reference

14: Order-3

14: Order-1

15: Reference

15: Order-2

15: Order-1

16: Reference

16: Order-2

16: Order-1

17: Reference

17: Order-2

17: Order-1

18: Reference

18: Order-3

18: Order-1

19: Reference

19: Order-3

19: Order-1

20: Reference

20: Order-5

20: Order-1

21: Reference

21: Order-5

21: Order-1

22: Reference

22: Order-5

22: Order-1

23: Reference

23: Order-5

23: Order-1

24: Reference

24: Order-3

24: Order-1

25: Reference

25: Order-5

25: Order-1

26: Reference

26: Order-4

26: Order-1

27: Reference

27: Order-4

27: Order-1

28: Reference

28: Order-5

28: Order-1

29: Reference

29: Order-5

29: Order-1

30: Reference

30: Order-4

30: Order-1

31: Reference

31: Order-4

31: Order-1

32: Reference

32: Order-4

32: Order-1

33: Reference

33: Order-4

33: Order-1

34: Reference

34: Order-4

34: Order-1

35: Reference

35: Order-1

35: Order-1

36: Reference

36: Order-2

36: Order-1

37: Reference

37: Order-4

37: Order-1

38: Reference

38: Order-3

38: Order-1

39: Reference

39: Order-4

39: Order-1

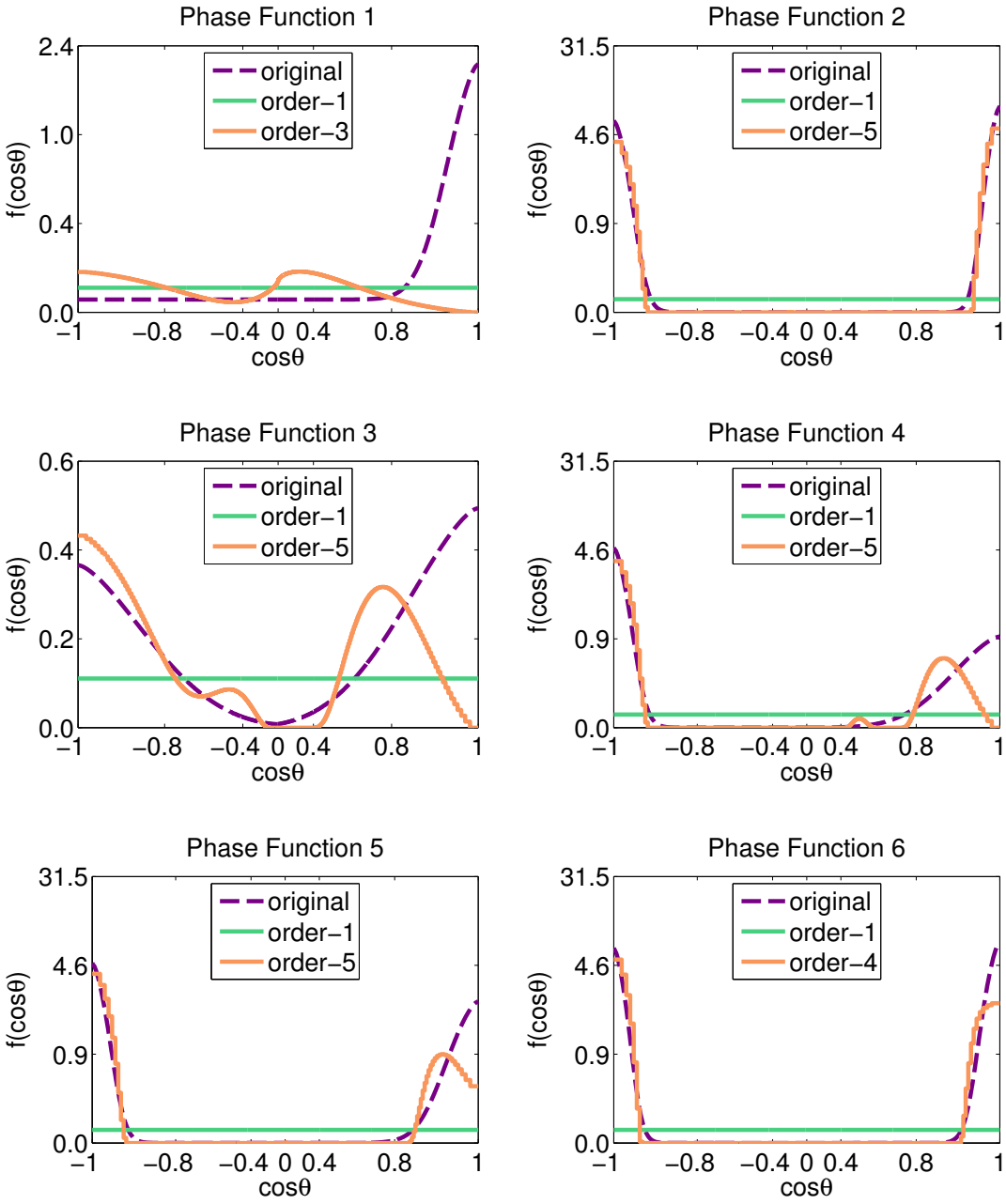
40: Reference

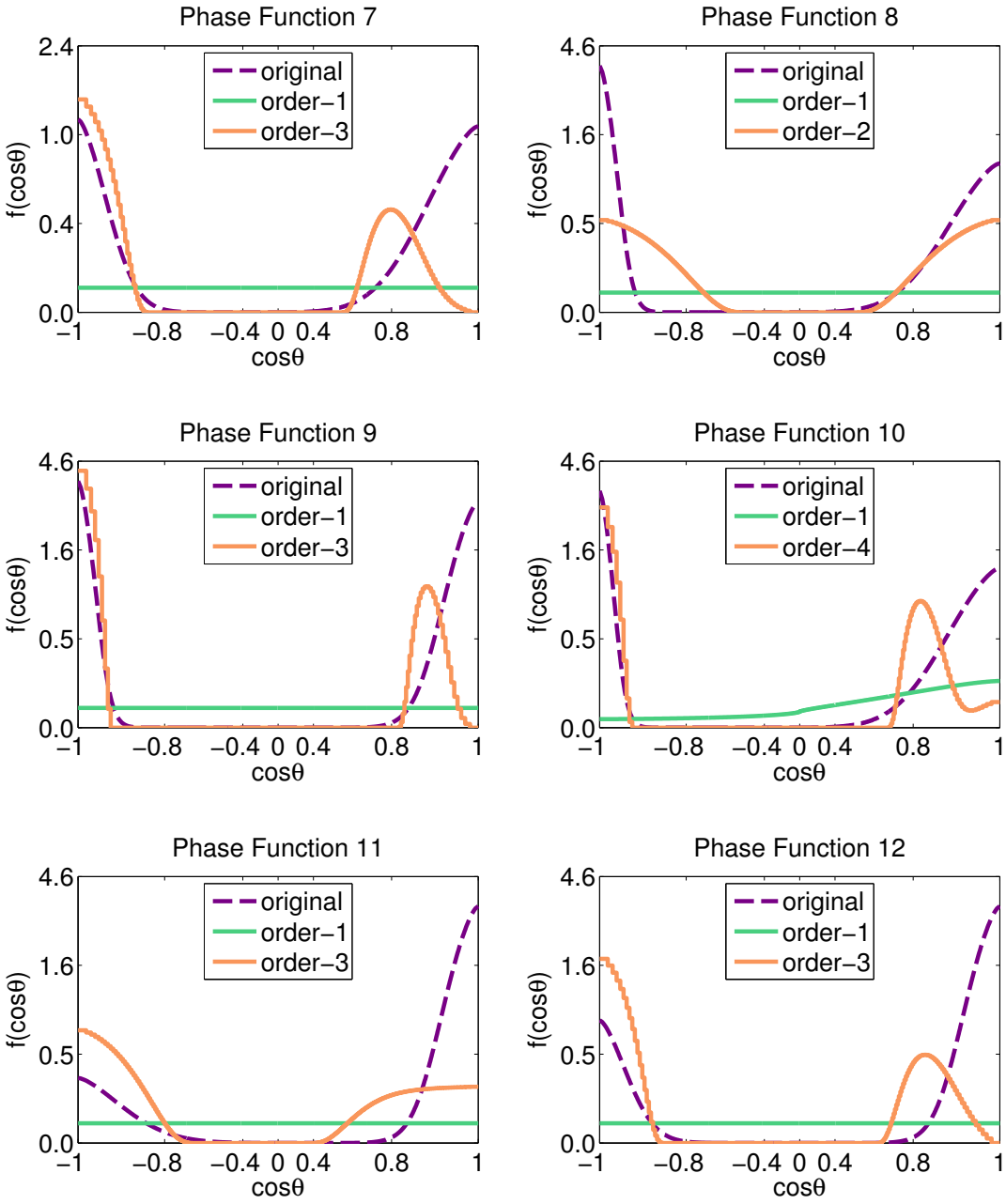
40: Order-4

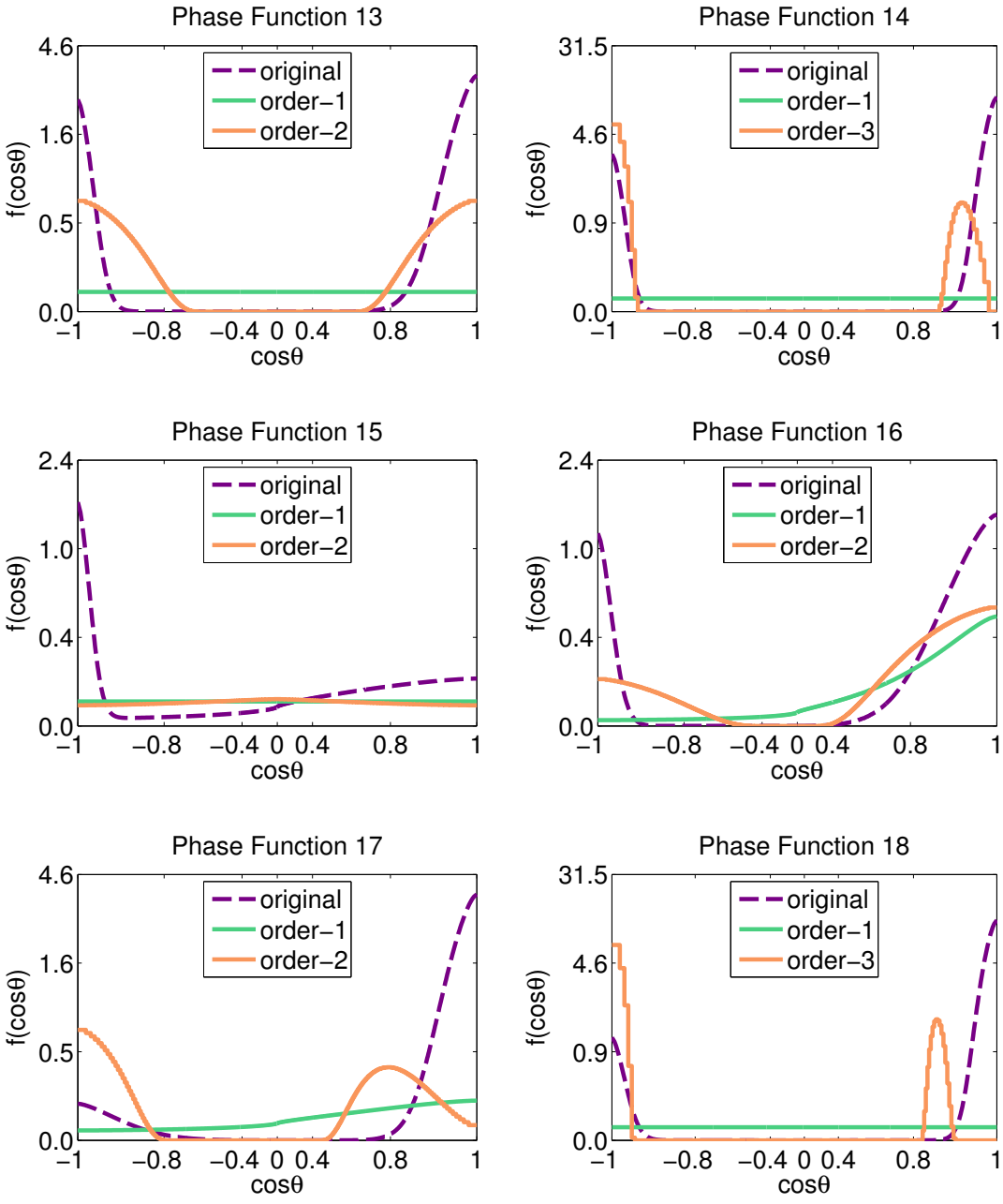
40: Order-1

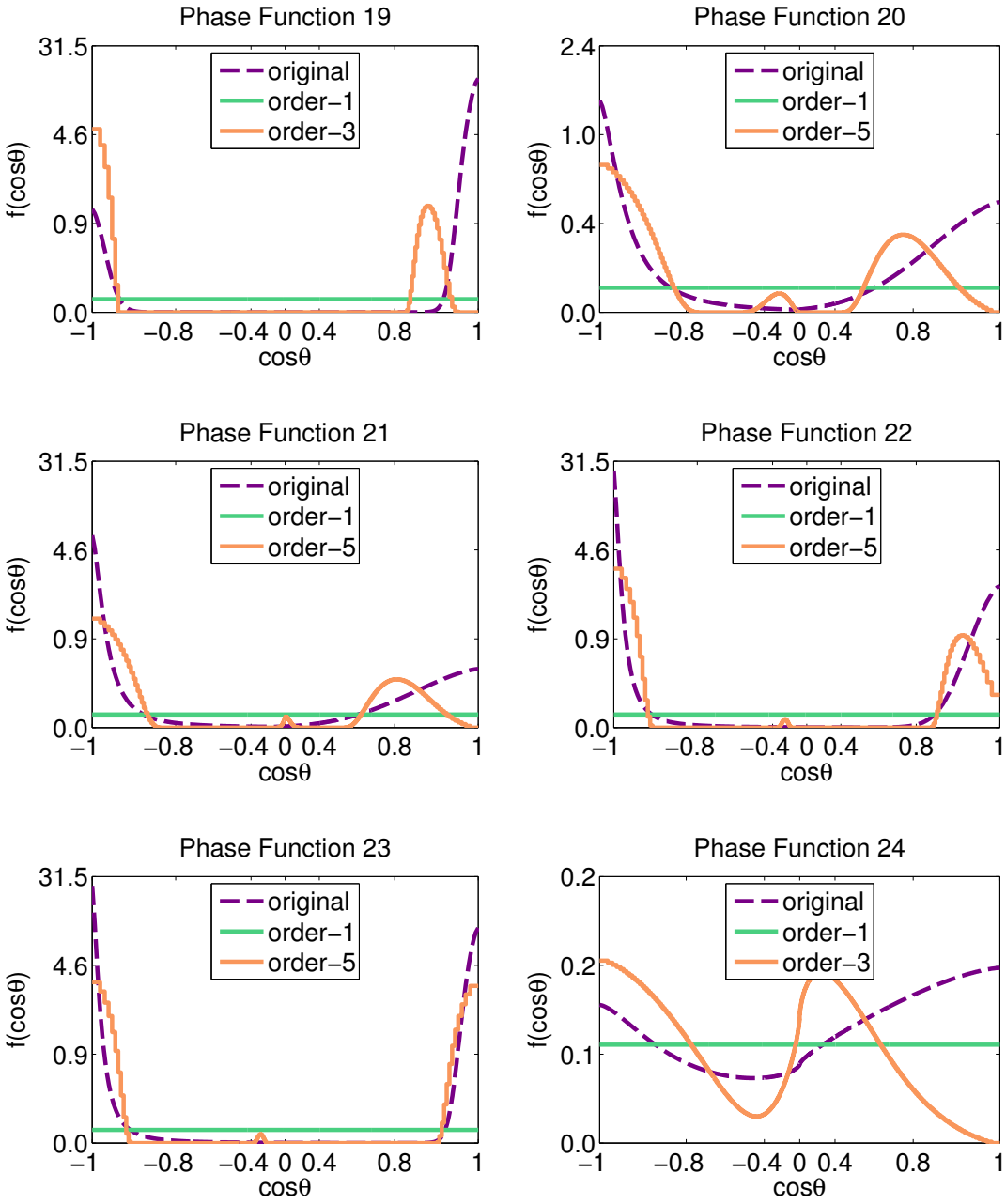
A.6.2 Phase function plots

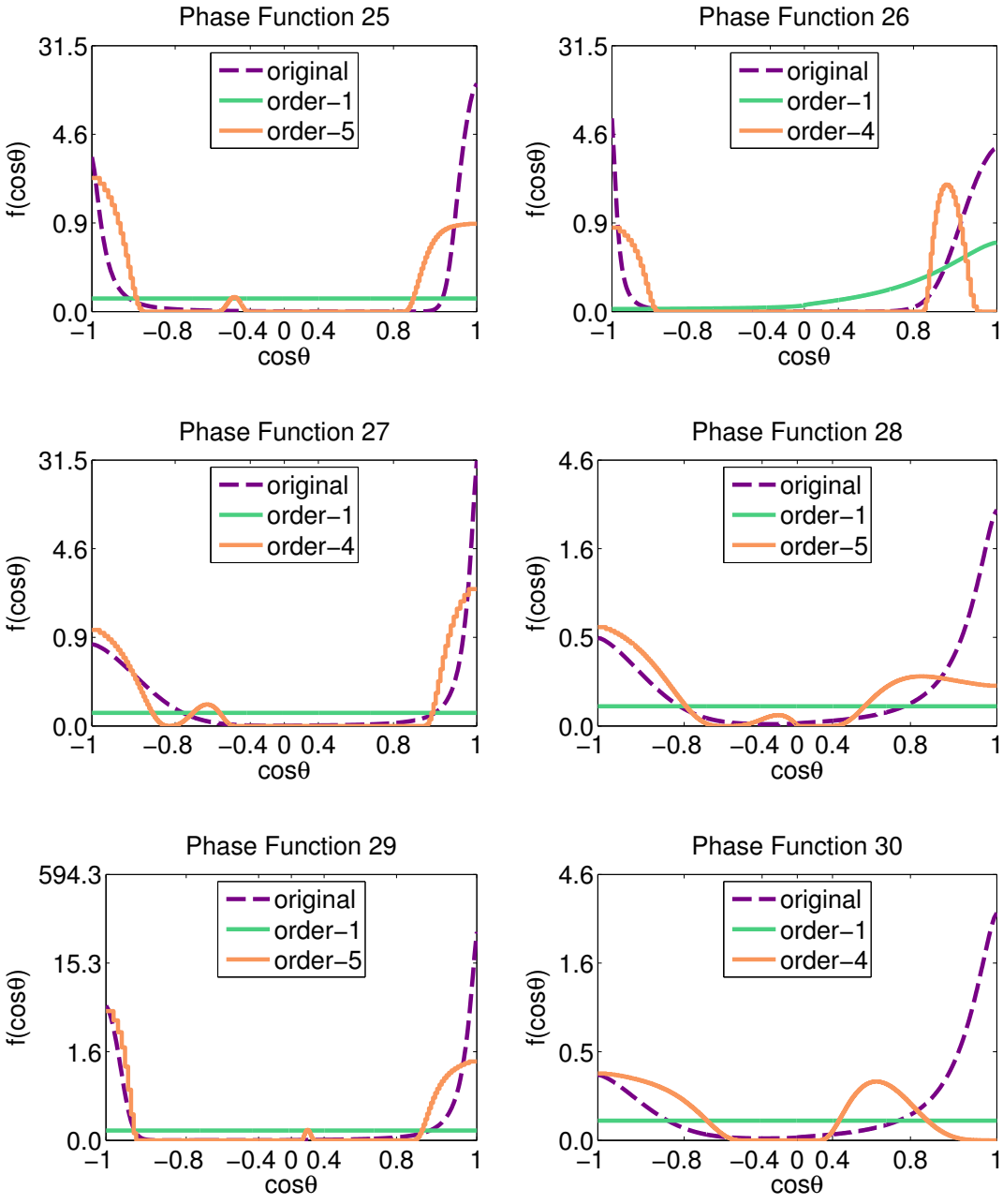
This section contains the plots of all phase functions used to create the renderings in Section A.6.1.

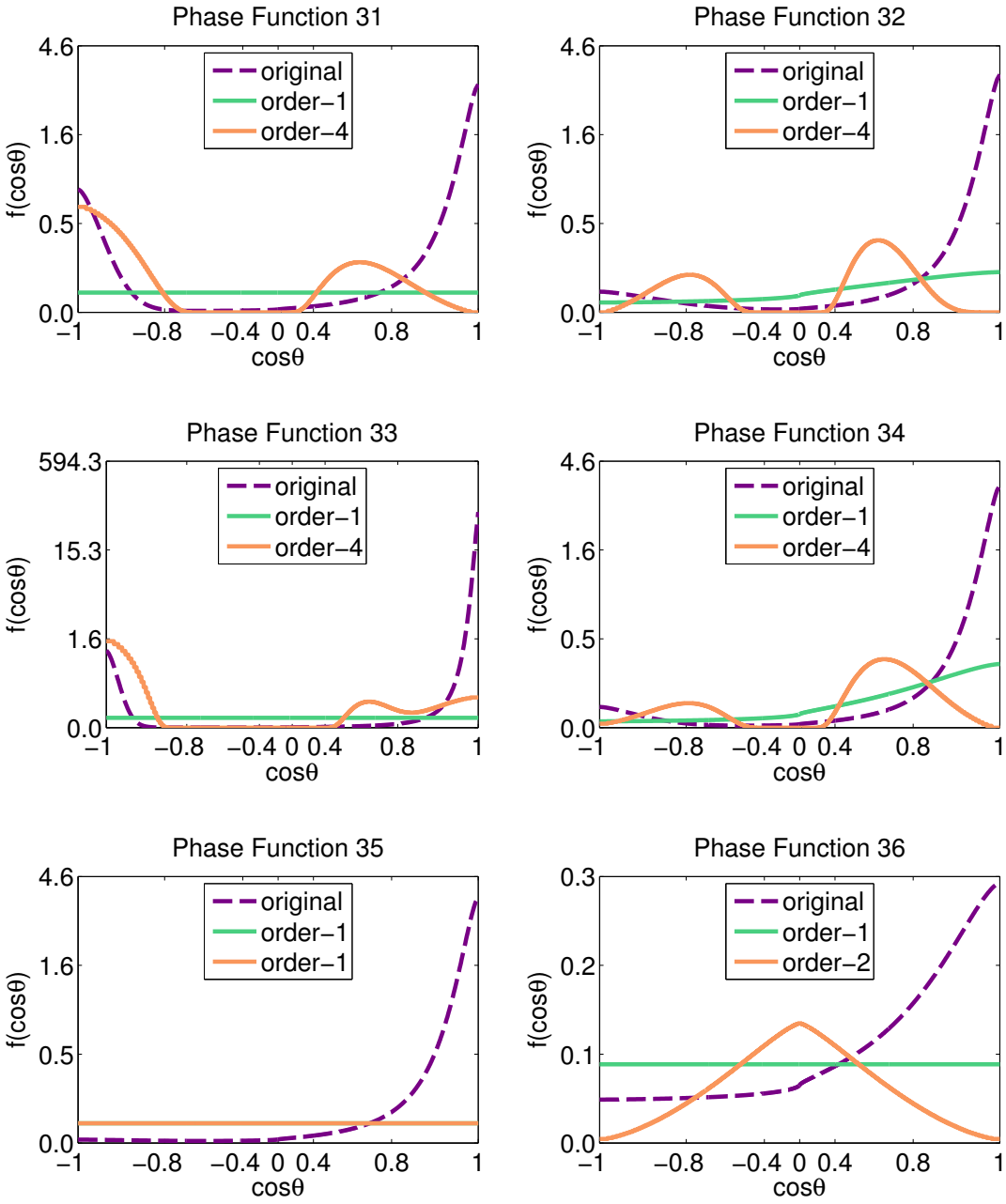

Phase Function 1
original
order−1
order−3
f(cosθ)
cosθ
Phase Function 2
original
order−1
order−5
f(cosθ)
cosθ
Phase Function 3
original
order−1
order−5
f(cosθ)
cosθ
Phase Function 4
original
order−1
order−5
f(cosθ)
cosθ
Phase Function 5
original
order−1
order−5
f(cosθ)
cosθ
Phase Function 6
original
order−1
order−4
f(cosθ)
cosθ

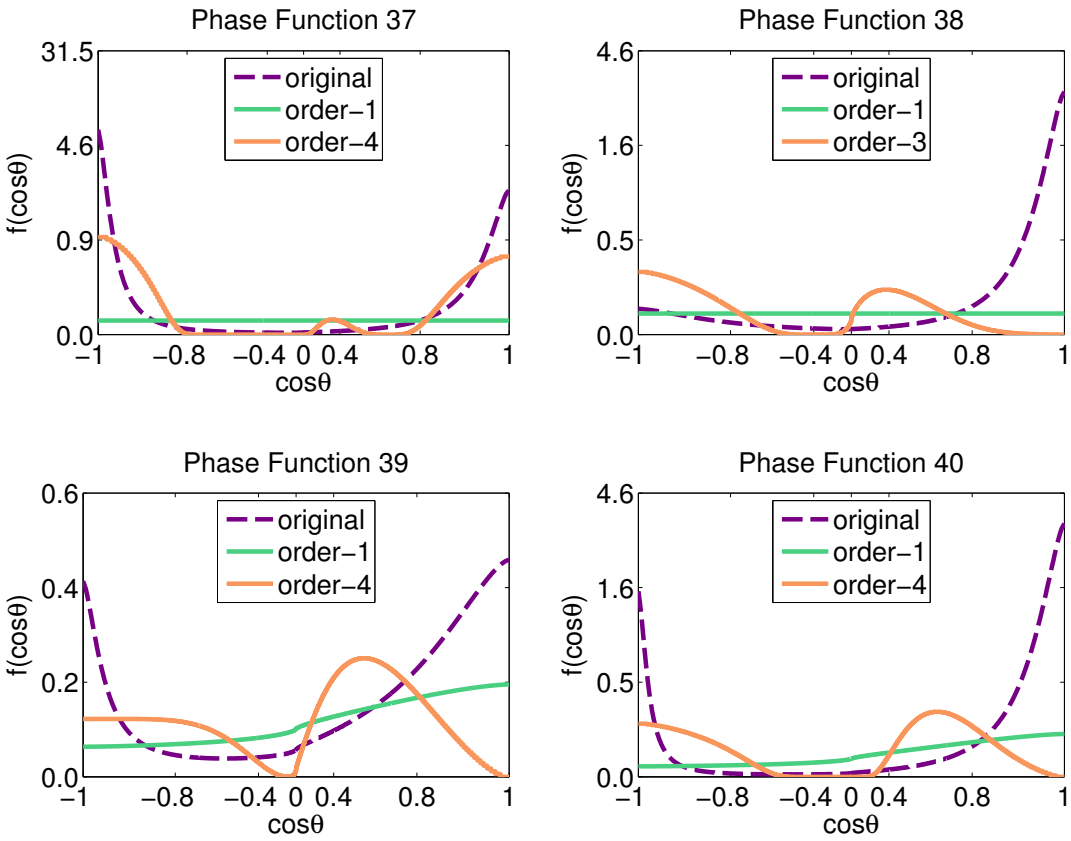

Phase Function 7
original
order−1
order−3
f(cosθ)
cosθ
Phase Function 8
original
order−1
order−2
f(cosθ)
cosθ
Phase Function 9
original
order−1
order−3
f(cosθ)
cosθ
Phase Function 10
original
order−1
order−4
f(cosθ)
cosθ
Phase Function 11
original
order−1
order−3
f(cosθ)
cosθ
Phase Function 12
original
order−1
order−3
f(cosθ)
cosθ


Phase Function 13
original
order−1
order−2
f(cosθ)
cosθ
4.6
1.6
0.5
0.0
−1
−0.8
−0.4 0 0.4
0.8
1
Phase Function 14
original
order−1
order−3
31.5
4.6
0.9
Phase Function 15
2.4
1.0
0.4
Phase Function 16
Phase Function 17
Phase Function 18


Phase Function 19
original
order−1
order−3
f(cosθ)
cosθ
31.5
4.6
0.9
0.0
−1
−0.8
−0.4 0 0.4
0.8
1
Phase Function 20
original
order−1
order−5
2.4
1.0
0.4
Phase Function 21
Phase Function 22
Phase Function 23
Phase Function 24
0.2
0.1


Phase Function 25
original
order−1
order−5
f(cosθ)
cosθ
31.5
4.6
0.9
0.0
−1
−0.8
−0.4 0 0.4
0.8
1
Phase Function 26
original
order−1
order−4
Phase Function 27
original
order−1
order−4
Phase Function 28
original
order−1
order−5
4.6
1.6
0.5
Phase Function 29
original
order−1
order−5
594.3
15.3
1.6
Phase Function 30
original
order−1
order−4


Phase Function 31
original
order−1
order−4
f(cosθ)
cosθ
4.6
1.6
0.5
0.0
−1
−0.8
−0.4
0
0.4
0.8
1
Phase Function 32
original
order−1
order−4
f(cosθ)
cosθ
Phase Function 33
original
order−1
order−4
f(cosθ)
cosθ
594.3
15.3
1.6
0.0
Phase Function 34
original
order−1
order−4
f(cosθ)
cosθ
Phase Function 35
original
order−1
order−1
f(cosθ)
cosθ
Phase Function 36
original
order−1
order−2
f(cosθ)
cosθ
0.3
0.2
0.1


Phase Function 37
31.5
4.6
0.9
0.0
f(cosθ)
original
order−1
order−4
−1
−0.8
−0.4 0 0.4
0.8
1
cosθ
Phase Function 38
4.6
1.6
0.5
0.0
f(cosθ)
original
order−1
order−3
−1
−0.8
−0.4 0 0.4
0.8
1
cosθ
Phase Function 39
0.6
0.4
0.2
0.0
f(cosθ)
original
order−1
order−4
−1
−0.8
−0.4 0 0.4
0.8
1
cosθ
Phase Function 40
4.6
1.6
0.5
0.0
f(cosθ)
original
order−1
order−4
−1
−0.8
−0.4 0 0.4
0.8
1
cosθ

BIBLIOGRAPHY

[1] Neeharika Adabala, Nadia Magnenat-Thalmann, and Guangzheng Fei. Visualization of woven cloth. In *14th Eurographics Workshop on Rendering*, pages 180–185, June 2003.

[2] Adam Arbree, Bruce Walter, and Kavita Bala. Heterogeneous subsurface scattering using the finite element method. *IEEE Trans. on Visualization and Computer Graphics*, 17(7):956–969, 2011.

[3] George Brown Arfken, Hans-Jurgen Weber, and Lawrence Ruby. *Mathematical methods for physicists*. Academic press New York, 1985.

[4] Michael Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[5] Michael Ashikhmin, Simon Premoze, and Peter S. Shirley. A microfacet-based brdf generator. In *Proceedings of ACM SIGGRAPH 2000*, pages 65–74, July 2000.

[6] M. Axelsson. Estimating 3D fibre orientation in volume images. In *International Conference on Pattern Recognition, 2008*, pages 1 –4, dec. 2008.

[7] Subrahmanyan Chandrasekhar. *Radiative transfer*. Courier Dover Publications, 1960.

[8] Stéphane Chatigny, Michel Morin, Daniel Asselin, Yves Painchaud, and Pierre Beaudry. Hybrid Monte Carlo for photon transport through optically thick scattering media. *Applied optics*, 38(28):6075–6086, 1999.

[9] Yanyun Chen, Stephen Lin an Hua Zhong, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Realistic rendering and animation of knitwear. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):43–55, 2003.

[10] Yanyun Chen, Xin Tong, Jiaping Wang, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Shell texture functions. *ACM Trans. Graph.*, 23(3):343–353, 2004.

[11] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. *ACM Trans. Graph.*, 22(3):287–294, 2003.

[12] Raúl E Curto and Lawrence A Fialkow. Recursiveness, positivity, and truncated moment problems. *Houston J. Math*, 17(4):603–635, 1991.

[13] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 18(1):1–34, 1999.

[14] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of SIGGRAPH 1998*, pages 189–198, 1998.

[15] Eugene D'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. *ACM Trans. Graph.*, 30(4):56:1–56:14, 2011.

[16] Yoshinori Dobashi, Wataru Iwasaki, Ayumi Ono, Tsuyoshi Yamamoto, Yonghao Yue, and Tomoyuki Nishita. An inverse problem approach for automatically adjusting the parameters for rendering clouds using photographs. *ACM Trans. Graph.*, 31(6):145:1–145:10, 2012.

[17] Yue Dong, Jiaping Wang, Xin Tong, John Snyder, Yanxiang Lan, Moshe Ben-Ezra, and Baining Guo. Manifold bootstrapping for SVBRDF capture. *ACM Trans. Graph.*, 29(4):98:1–98:10, 2010.

[18] Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. *ACM Trans. Graph.*, 24(3):1032–1039, 2005.

[19] Craig Donner and Henrik Wann Jensen. Rendering translucent materials using photon diffusion. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 243–251. Eurographics Association, 2007.

[20] Frédéric Drago and Norishige Chiba. Painting canvas synthesis. *The Visual Computer*, 20(5):314–328, 2004.

[21] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038. IEEE, 1999.

[22] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 341–346, 2001.

[23] Raanan Fattal. Participating media illumination using light propagation maps. *ACM Trans. Graph.*, 28(1):7:1–7:11, 2009.

[24] G.E. Forsythe and R.A. Leibler. Matrix inversion by a monte carlo method. *Mathematical Tables and Other Aids to Computation*, pages 127–129, 1950.

[25] Jeppe Revall Frisvad, Niels Jørgen Christensen, and Henrik Wann Jensen. Computing the scattering properties of participating media using Lorenz-Mie theory. *ACM Trans. Graph.*, 26(3):60:1–60:10, 2007.

[26] R. Furukawa, H. Kawasaki, K. Ikeuchi, and M. Sakauchi. Appearance based object modeling using texture database: acquisition, compression and rendering. In *Eurographics Workshop on Rendering*, pages 257–266, 2002.

[27] Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. Estimating specular roughness and anisotropy from second order spherical gradient illumination. *Comput. Graph. Forum*, 28(4):1161–1170, 2009.

[28] Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. Circularly polarized spherical illumination reflectometry. *ACM Trans. Graph.*, 29(6):162:1–162:12, 2010.

[29] Ioannis Gkioulekas, Bei Xiao, Shuang Zhao, Edward H. Adelson, Todd Zickler, and Kavita Bala. Understanding the role of phase function in translucent appearance. *ACM Trans. Graph.*, 32(5):147:1–147:19, 2013.

[30] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Trans. Graph.*, 32(6), November 2013.

[31] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Trans. Graph.*, 32(6):162:1–162:13, 2013.

[32] R.H Gong, B Ozgen, and M Soleimani. Modeling of yarn cross-section in plain woven fabric. *Textile Research Journal*, 79(11):1014–1020, 2009.

[33] Gurobi. Gurobi optimization libraries, 2013. www.gurobi.com.

[34] Ralf Habel, Per H Christensen, and Wojciech Jarosz. Photon beam diffu-

sion: A hybrid monte carlo method for subsurface scattering. In *Computer Graphics Forum*, volume 32, pages 27–37, 2013.

[35] Miloš Hašan, Martin Fuchs, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. Physical reproduction of materials with specified subsurface scattering. *ACM Trans. Graph.*, 29(4):61:1–61:10, 2010.

[36] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH 1995*, pages 229–238, 1995.

[37] Louis G Henyey and Jesse L Greenstein. Diffuse radiation in the galaxy. *The Astrophysical Journal*, 93:70–83, 1941.

[38] T. Hurtut, P.-E. Landes, J. Thollot, Y. Gousseau, R. Drouillhet, and J.-F. Coeurjolly. Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, pages 51–60, 2009.

[39] Piti Irawan and Steve Marschner. Specular reflection from woven cloth. *ACM Trans. Graph.*, 31(1):11:1–11:20, 2012.

[40] Akira Ishimaru. *Wave propagation and scattering in random media*, volume 2. Academic press New York, 1978.

[41] Wenzel Jakob. *Light transport on path-space manifolds*. PhD thesis, Cornell University, 2013.

[42] Wenzel Jakob. Mitsuba renderer, 2014. www.mitsuba-renderer.org.

[43] Wenzel Jakob, Adam Arbree, Jonathan T. Moon, Kavita Bala, and Steve Marschner. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph.*, 29(4):53:1–53:13, 2010.

[44] Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Trans. Graph.*, 30(1):5:1–5:19, 2011.

[45] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. AK Peters, Ltd., 2001.

[46] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scences with participating media using photon maps. In

Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pages 311–320, 1998.

[47] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001*, pages 511–518, 2001.

[48] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. *SIGGRAPH Computer Graphics*, 23(3):271–280, 1989.

[49] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, pages 271–280, July 1989.

[50] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18:165–174, 1984.

[51] Jonathan Kaldor. *Simulating yarn-based cloth*. PhD thesis, Cornell University, 2011.

[52] S. Kawabata, M. Niwa, and H. Kawai. The finite deformation theory of plain weave fabrics. part i: The biaxial deformation theory. *Journal of Textile Instiute*, 64(1):21–46, 1973.

[53] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. *ACM Trans. Graph.*, 26(3), 2007.

[54] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2D exemplars. *ACM Trans. Graph.*, 26(3):2:1–2:9, 2007.

[55] Mikhail K Kozlov, Sergei P Tarasov, and Leonid G Khachiyan. The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5):223–228, 1980.

[56] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Trans. Graph.*, 24(3):795–802, 2005.

[57] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick.

Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, 2003.

[58] Eric P Lafortune and Yves D Willems. Rendering participating media with bidirectional path tracing. In *Rendering Techniques 96*, pages 91–100. Springer, 1996.

[59] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Trans. Graph.*, 24(3):777–786, 2005.

[60] Hongsong Li, Fabio Pellacini, and Kenneth E Torrance. A hybrid Monte Carlo method for accurate and efficient subsurface scattering. In *Proceedings of EGSR 2005*, pages 283–290, 2005.

[61] S.V Lomov, R.S Parnas, S Bandyopadhyay Ghosh, I Verpoest, and A Nakai. Experimental and theoretical characterization of the geometry of two-dimensional braided fabrics. *Textile Research Journal*, 72(8):706–712, 2002.

[62] Bradford J. Loos, Lakulish Antani, Kenny Mitchell, Derek Nowrouzezahrai, Wojciech Jarosz, and Peter-Pike Sloan. Modular radiance transfer. *ACM Trans. Graph.*, 30(6):178:1–178:10, 2011.

[63] R. Lu, J. J. Koenderink, and A. M. L. Kappers. Optical properties (bidirectional reflection distribution functions) of velvet. *Applied Optics*, 37(25):5974–5984, 1998.

[64] Chongyang Ma, Li-Yi Wei, and Xin Tong. Discrete element textures. *ACM Trans. Graph.*, 30(4):62:1–62:10, 2011.

[65] Sebastian Magda and David Kriegman. Reconstruction of volumetric surface textures for real-time rendering. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)*, pages 19–29, 2006.

[66] Rafat Mantiuk, Kil Joong Kim, Allan G. Rempel, and Wolfgang Heidrich. HDR-VDP-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions. *ACM Trans. Graph.*, 30(4):40:1–40:14, 2011.

[67] Stephen R. Marschner, Stephen H. Westin, Adam Arbree, and Jonathan T. Moon. Measuring and modeling the appearance of finished wood. *ACM Trans. Graph.*, 24(3):727–734, 2005.

[68] Stephen R. Marschner, Stephen H. Westin, Adam Arbree, and Jonathan T. Moon. Measuring and modeling the appearance of finished wood. *ACM Trans. Graph.*, 24(3):727–734, 2005.

[69] Paul Merrell and Dinesh Manocha. Continuous model synthesis. *ACM Trans. Graph.*, 27(5):158:1–158:7, 2008.

[70] Jonathan T. Moon and Stephen R. Marschner. Simulating multiple scattering in hair using a photon mapping approach. *ACM Trans. Graph.*, 25:1067–1074, 2006.

[71] Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner. Rendering discrete random media using precomputed scattering solutions. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 231–242. Eurographics Association, 2007.

[72] Jonathan T. Moon, Bruce Walter, and Steve Marschner. Efficient multiple scattering in hair using spherical harmonics. *ACM Trans. Graph.*, 27(3):31:1–31:7, 2008.

[73] Isamu Motoyoshi, Shinya Nishida, Lavanya Sharan, and Edward Adelson. Image statistics and the perception of surface qualities. Nature, pages 206–209, 2007.

[74] Srinivasa G. Narasimhan and Shree K. Nayar. Shedding light on the weather. In *Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition*, pages 665–672. IEEE Computer Society, 2003.

[75] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of BRDF models. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering*, pages 117–126, June 2005.

[76] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. Virtual ray lights for rendering scenes with participating media. *ACM Trans. Graph.*, 31(4):60:1–60:11, 2012.

[77] Marios Papas, Christian Regg, Wojciech Jarosz, Bernd Bickel, Philip Jackson, Wojciech Matusik, Steve Marschner, and Markus Gross. Fabricating translucent materials using continuous pigment mixtures. *ACM Trans. Graph.*, 32(4):146:1–146:12, 2013.

[78] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis light transport for participating media. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 11–22, 2000.

[79] K. Perlin and E. M. Hoffert. Hypertexture. *SIGGRAPH Comput. Graph.*, 23(3):253–262, 1989.

[80] F. T. Pierce. The geometry of cloth structure. *Journal of the Textile Institute*, 28(3):45–96, 1937.

[81] Pointcarré. Pointcarré textile software, 2001. pointcarre.com.

[82] Gerald C Pomraning. *The equations of radiation hydrodynamics*. Courier Dover Publications, 1973.

[83] Serban D. Porumbescu, Brian Budge, Louis Feng, and Kenneth I. Joy. Shell maps. *ACM Trans. Graph.*, 24(3):626–633, July 2005.

[84] Simon Premože, Michael Ashikhmin, Jerry Tessendorf, Ravi Ramamoorthi, and Shree Nayar. Practical rendering of multiple scattering effects in participating media. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 363–374. Eurographics Association, 2004.

[85] Ganesh Ramanarayanan and Kavita Bala. Constrained texture synthesis via energy minimization. *IEEE Trans. on Visualization and Computer Graphics*, 13(1):167–178, 2007.

[86] D. Rolfsen. *Knots and links*. American Mathematical Society, 2003.

[87] Holly E. Rushmeier and Kenneth E. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. *SIGGRAPH Comput. Graph.*, 21(4):293–302, 1987.

[88] Iman Sadeghi, Oleg Bisker, Joachim De Deken, and Henrik Wann Jensen. A practical microcylinder appearance model for cloth rendering. *ACM Trans. Graph.*, 32(2):14:1–14:12, 2013.

[89] K. Schroder, Reinhard Klein, and Arno Zinke. A volumetric approach to predictive rendering of fabrics. *Comput. Graph. Forum*, 30(4):1277–1286, 2011.

[90] T Shinohara, J Takayama, S Ohyama, and A Kobayashi. Extraction of yarn positional information from a three-dimensional CT image of textile fabric using yarn tracing with a filament model for structure analysis. *Textile Research Journal*, 80(7):623–630, 2010.

[91] Jos Stam. Multiple scattering as a diffusion process. In *Rendering Techniques 95*, pages 41–50. 1995.

[92] X Thibault and J Bloch. Structural analysis by X-ray microtomography of a strained nonwoven papermaker felt. *Textile Research Journal*, 72(6):480–485, 2002.

[93] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. Graph.*, 21(3):665–672, 2002.

[94] Gert Van den Eynde. *Neutron Transport with Anisotropic Scattering*. PhD thesis, Ph. D. Thesis, Université Libre de Bruxelles, 2005.

[95] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, 1997.

[96] B. Walter, S.R. Marschner, H. Li, and K.E. Torrance. Microfacet models for refraction through rough surfaces. In *Eurographics Symposium on Rendering*, pages 195–206, 2007.

[97] Bruce Walter, Pramook Khungurn, and Kavita Bala. Bidirectional lightcuts. *ACM Trans. Graph.*, 31(4):59:1–59:11, 2012.

[98] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of EGSR 2007*, pages 195–206, 2007.

[99] Jiaping Wang, Shuang Zhao, Xin Tong, Stephen Lin, Zhouchen Lin, Yue Dong, Baining Guo, and Heung-Yeung Shum. Modeling and rendering of heterogeneous translucent materials using the diffusion equation. *ACM Trans. Graph.*, 27(1):9:1–9:18, 2008.

[100] Jiaping Wang, Shuang Zhao, Xin Tong, John Snyder, and Baining Guo. Modeling anisotropic surface reflectance with example-based microfacet synthesis. *ACM Trans. Graph.*, 27(3):41:1–41:9, 2008.

[101] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 479–488, 2000.

[102] Stephen H. Westin, James R. Arvo, and Kenneth E. Torrance. Predicting reflectance functions from complex surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 255–264, July 1992.

[103] E Woodcock, T Murphy, P Hemmings, and S Longworth. Techniques used in the GEM code for monte carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conf. Applications of Computing Methods to Reactor Problems*, volume 557, 1965.

[104] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Trans. Graph.*, 23(3):364–367, 2004.

[105] Douglas R Wyman, Michael S Patterson, and Brian C Wilson. Similarity relations for anisotropic scattering in Monte Carlo simulations of deeply penetrating neutral particles. *Journal of Computational Physics*, 81(1):137–150, 1989.

[106] Douglas R Wyman, Michael S Patterson, and Brian C Wilson. Similarity relations for the interaction parameters in radiation transport. *Applied optics*, 28(24):5243–5249, 1989.

[107] Ying-Qing Xu, Yanyun Chen, Stephen Lin, Hua Zhong, Enhua Wu, Baining Guo, and Heung-Yeung Shum. Photorealistic rendering of knitwear using the Lumislice. In *Proceedings of SIGGRAPH 2001*, pages 391–398, 2001.

[108] Ying-Qing Xu, Yanyun Chen, Stephen Lin, Hua Zhong, Enhua Wu, Baining Guo, and Heung-Yeung Shum. Photorealistic rendering of knitwear using the lumislice. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 391–398, 2001.

[109] Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi, and Tomoyuki Nishita. Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. *ACM Trans. Graph.*, 29(6):177:1–177:8, 2010.

[110] Shuang Zhao, Miloš Hašan, Ravi Ramamoorthi, and Kavita Bala. Modular flux transfer: Efficient rendering of high-resolution volumes with repeated structures. *ACM Trans. Graph.*, 32(4):131:1–131:12, 2013.

[111] Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. Building volumetric appearance models of fabric using micro CT imaging. *ACM Trans. Graph.*, 30(4):44:1–44:10, 2011.

[112] Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. Structure-aware synthesis for predictive woven fabric appearance. *ACM Trans. Graph.*, 31(4):75:1–75:10, 2012.

[113] Shuang Zhao, Ravi Ramamoorthi, and Kavita Bala. High-order similarity relations in radiative transfer. *ACM Trans. Graph.*, 33(4), 2014.

[114] Kun Zhou, Xin Huang, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. Mesh quilting for geometric texture synthesis. *ACM Trans. Graph.*, 25(3):690–697, 2006.

[115] Arno Zinke, Cem Yuksel, Andreas Weber, and John Keyser. Dual scattering approximation for fast multiple scattering in hair. *ACM Trans. Graph.*, 27(3):32:1–32:10, 2008.