Downsampling Scattering Parameters for Rendering Anisotropic Media

Shuang Zhao[1] * Lifan Wu[2] * Frédo Durand[3] Ravi Ramamoorthi[2]
[1]University of California, Irvine [2]University of California, San Diego [3]MIT

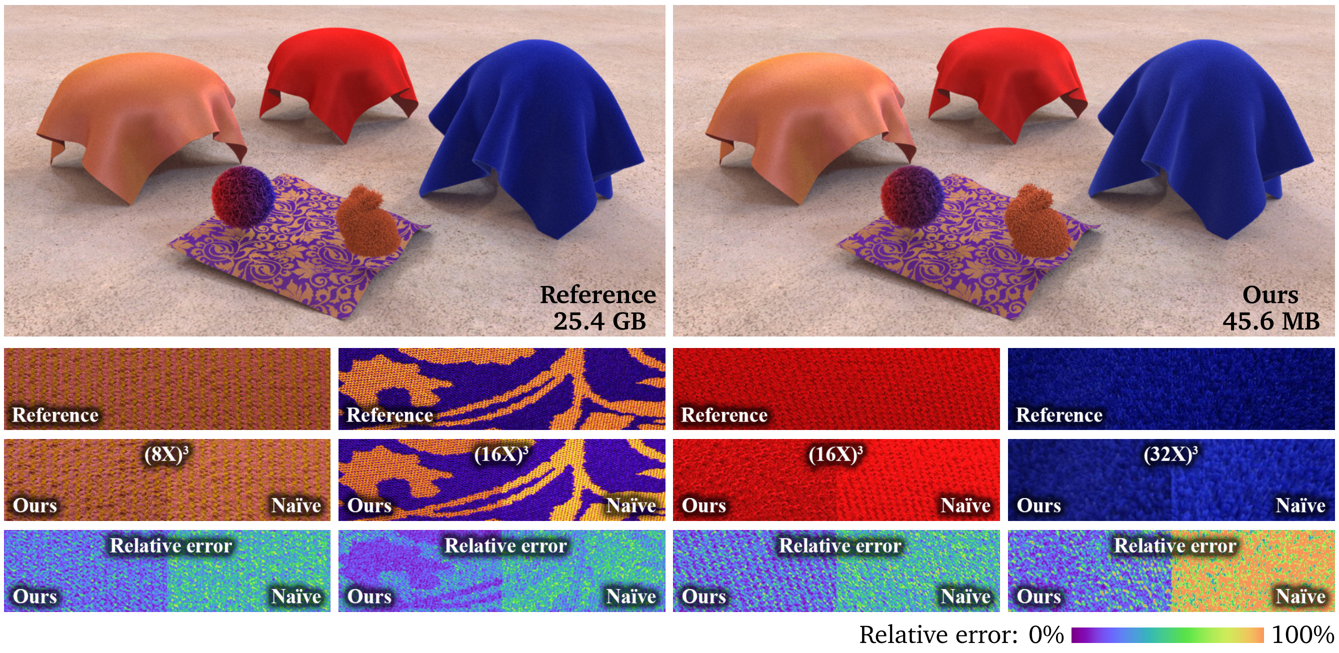


Figure 1: *We present a new approach to compute scattering parameters at reduced resolutions. Many detailed appearance models involve high-resolution volumetric representations (top-left). Such level of detail leads to high storage but is usually unnecessary especially when the object is rendered at a distance. However, naïve downsampling often loses intrinsic shadowing structures and brightens resulting images (see the insets). Our method computes scaled phase functions, a combined representation of single-scattering albedo and phase function, and provides significantly better accuracy while reducing the data size by almost three orders of magnitude (top-right).*

Abstract

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

Keywords: radiative transfer, multi-resolution, level of details, pre-filtering, global illumination

Concepts: •**Computing methodologies** → **Rendering;**

*Joint first authors.

1 Introduction

Recently, detailed volumetric models [Jakob et al. 2010; Zhao et al. 2011] have become increasingly more popular for handling the appearance of materials with complex 3D structures (e.g., fabric and fur). These *micro-appearance* models explicitly capture objects' microscopic level geometries and have brought the quality of computer rendered complex materials to the next level.

Unfortunately, efficient use of these high-resolution models remains challenging since they are usually highly data intensive, and loading these data into memory alone can take minutes. For many computer graphics applications, however, such extreme high-resolution is unnecessary: especially when the object is viewed from a distance. Greatly downsampled versions, which require significantly less storage, would suffice in many situations. In Figure 1, for instance, the reference scene involves 4.68×10^{12} effective voxels taking 25.4 GB of storage. Our models with reduced resolutions (see the second row of insets for levels of downsampling), on the other hand, result in only 45.6 MB of data.

Although lower-resolution operations have been developed for plain textures and normal maps [Han et al. 2007], they are

SA '16 Technical Papers, December 05-08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2980228

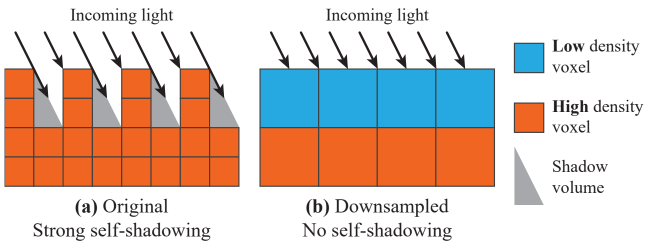


Figure 2: *2D demonstration of the weakening of self-occlusion caused by downsampling of spatially varying densities.*

largely lacking for light scattering parameters [Chandrasekhar 1960; Jakob et al. 2010]. Recently, Heitz et al. [2015] proposed a new phase function formulation which offers easy (trilinear) prefiltering and is a valuable first-step toward this direction. However, linear downsampling is generally insufficient when handling scattering parameters including spatially varying densities and albedo values. This is because downsampling weakens intrinsic shadowing structures (Figure 2), causing significant brightening of resulting renderings (see the middle row of insets in Figure 1).

In this paper, we introduce a novel method to improve the accuracy of downsampled scattering parameters. In particular, our approach optimizes single-scattering albedos and phase functions jointly. Our contributions include:

- We introduce *scaled phase functions* combining albedos and phase functions (§4).
- We develop an optimization based method to *downsample* scaled phase functions (§5, §6).
- We show how *modularity* can be exploited by reusing a single set of optimized parameters for multiple objects, significantly reducing the amortized optimization overhead (§7).

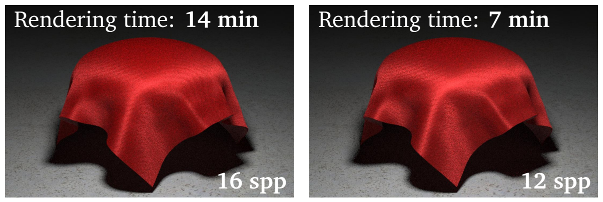
Our approach is model-dependent: unlike traditional prefiltering methods, our technique takes full input geometries into consideration. Our optimization involves a few training renderings which can take tens of CPU core hours but is comparable to rendering one high-quality image. Fortunately, this process only needs to be executed once: the resulting representations can be reused in different virtual configurations. Furthermore, by exploiting modularity, multiple objects can share one single optimization. We also intend to release all our downsampled datasets so that the detailed models in Figures 1, 18, and 19 can be directly used by other researchers.

Our downsampled models offer several orders of magnitude storage saving and greatly reduce the I/O time, which can be a significant overhead for highly detailed scenes like Figure 1. In addition, our models are less prone to aliasing and usually require fewer sample paths to achieve similar rendering qualities (Figure 3). Lastly, these benefits are "effort-free": our technique allows the user to replace original parameters using significantly downsampled versions without modifying the core rendering algorithm.

2 Related Work

Radiative transfer. The scattering parameters considered in this paper arise from radiative transfer [Chandrasekhar 1960] which originally assumes random media with disorganized microstructures. Jakob et al. [2010] later relaxed this assumption by modeling anisotropic media with structured micro-geometries. [Heitz et al. 2015] has recently proposed a new anisotropic phase function offering fast evaluation and easy (linear) interpolation. We build our framework upon this state-of-the-art representation.

Micro-appearance models. Recently, a family of *micro-appearance modeling techniques* [Zhao et al. 2011; Zhao et al.



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

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

Figure 3: *Equal-quality renderings using volumes at original (a) and $(16\times)^3$-downsampled (b) resolutions (computed with our method). The lower-resolution representation requires fewer samples per pixel to obtain a similar resulting noise level (measured using [Liu et al. 2012]).*

2012] have been introduced. These methods represent material geometries at unprecedented detail using ultra high-resolution volumes. Leveraging anisotropic radiative transfer, they have provided extremely high-quality renderings. However, the use of large 3D volumes has yielded multi-gigabyte models. This paper focuses on greatly reducing the resolution of these models while maintaining good accuracy (Figures 1, 18, 19).

Handling high-resolution data/appearance. Several techniques (e.g., [Kraus and Bürger 2008; Sicat et al. 2014]) have been developed previously to interpolate voxelized data for efficient 3D visualization. Although these approaches could be adapted to handle some scattering parameters, they normally offer limited accuracy. In addition, a number of techniques have been introduced to efficiently handle surface-based reflectance profiles (e.g., [Tan et al. 2005; Bruneton and Neyret 2012; Jarabo et al. 2014]), micro-geometry (e.g., [Westin et al. 1992; Cook et al. 2007; Jakob et al. 2014]), or specialized volumes [Meng et al. 2015]. Unfortunately, these methods are not easily applicable to our problem.

Altering material properties. This paper aims to compute a material's scattering parameters at reduced resolutions. Previously, a family of techniques focusing on altering material scattering parameters are similarity relations [Zhao et al. 2014]. These methods, however, need to be applied independently to each point and generally do not allow changing model resolutions. Our technique is completely orthogonal and complementary to them.

Inverse rendering. Inverse rendering techniques solve for material optical parameters given desired object appearance. Some of them (e.g., [Gkioulekas et al. 2013; Hašan and Ramamoorthi 2013; Khungurn et al. 2015]) are also based on optimizations. However, these methods effectively solve a special case of our problem. That is, they assume homogeneity (i.e., spatially invariant parameters) and/or simple (single-lobe) phase functions. Our approach performs a joint optimization of both albedo and phase function, and supports heterogeneity.

3 Background

At the core of the radiative transfer framework [Chandrasekhar 1960] lies the *radiative transfer equation* (RTE)[1]

$$(\omega \cdot \nabla)L(\omega) = -\sigma_t L(\omega) + \sigma_s \int_{\mathbb{S}^2} L(\omega')f(\omega' \to \omega)\,\mathrm{d}\omega', \quad (1)$$

where σ_t (extinction coefficient), σ_s (scattering coefficient), and f (phase function) are *material scattering parameters*, and L is

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

the resulting radiance field. In addition, *single-scattering albedo* is defined as the ratio between σ_s and σ_t: $\alpha = \sigma_s/\sigma_t$. Table 1 summarizes all symbols commonly used in this paper.

Jakob et al. [2010] generalized the RTE Eq. (1) to capture directional dependencies of the scattering parameters, yielding the *anisotropic* RTE

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

Under this formulation, the medium is modeled as a collection of two-sided small mirrors, or *micro-flakes*, whose normal directions $\boldsymbol{m}$ follow a statistical distribution D. Then, the scattering parameters in Eq. (2) are given by $\sigma_t(\boldsymbol{\omega}) = \rho\sigma(\boldsymbol{\omega})$, $\sigma_s(\boldsymbol{\omega}) = \alpha\sigma_t(\boldsymbol{\omega})$,

$$f(\boldsymbol{\omega}'\to\boldsymbol{\omega}) = \frac{1}{\sigma(\boldsymbol{\omega}')}\int_{\mathbb{S}^2} p(\boldsymbol{m},\boldsymbol{\omega}'\to\boldsymbol{\omega})\langle\boldsymbol{\omega}',\boldsymbol{m}\rangle D(\boldsymbol{m})\,\mathrm{d}\boldsymbol{m}, \tag{3}$$

where ρ indicates the density of micro-flakes, α denotes the directionally independent albedo, p is the scattering profile for each micro-flake, and $\sigma(\boldsymbol{\omega}) = \int_{\mathbb{S}^2}\langle\boldsymbol{\omega},\boldsymbol{\omega}'\rangle D(\boldsymbol{\omega}')\,\mathrm{d}\boldsymbol{\omega}'$ provides projected area of the micro-flakes in direction $\boldsymbol{\omega}$. This generalized framework collapses to the standard isotropic case (1) when D is set to uniform [Jakob et al. 2010]. In addition, it has been shown to be capable of closely capturing the appearance of complex materials such as fabrics and fur [Jakob et al. 2010; Zhao et al. 2011; Zhao et al. 2012; Heitz et al. 2015].

Recently, Heitz et al. [2015] introduced a SGGX-based representation for the micro-flake normal distribution:

$$D(\boldsymbol{m}) := \frac{1}{\pi\sqrt{|S|}(\boldsymbol{m}^T S^{-1}\boldsymbol{m})^2}, \tag{4}$$

where S is a 3×3 symmetric positive definite matrix. This distribution has been demonstrated to offer similar representative power as state-of-the-art models [Jakob et al. 2010; Zhao et al. 2011] while being less expensive computationally. In the rest of this paper, we use $f^{\text{SGGX}}(S)$ to denote the SGGX phase function determined by S via Eq. (3) and Eq. (4).

Interpolation of SGGX phase function. To represent the average of n SGGX phase functions determined by matrices $S_1, S_2, \ldots, S_n$ with one SGGX function with the matrix $\tilde{S}$, Heitz et al. [2015] demonstrated that using the arithmetic average of those matrices, namely setting

$$\tilde{S} = \frac{1}{n}\left(\sum_{i=1}^{n} S_i\right), \tag{5}$$

can lead to a good approximation.

Interpolation of density and albedo. Previously, Kraus and Bürger [2008] demonstrated that independently interpolating (i.e., averaging) density and single-scattering albedo generally yields very poor results. Instead, given n extinction coefficients $\sigma_{t,1}^{\text{orig}}, \ldots, \sigma_{t,n}^{\text{orig}}$ and corresponding albedo $\alpha_1^{\text{orig}}, \ldots, \alpha_n^{\text{orig}}$, it is better to set the interpolated extinction coefficient $\bar{\sigma}_t$ and albedo $\bar{\alpha}$ using:

$$\bar{\sigma}_t = \frac{1}{n}\sum_{i=1}^{n}\sigma_{t,i}^{\text{orig}} \quad\text{and}\quad \bar{\alpha} = \frac{\sum_{i=1}^{n}\sigma_{t,i}^{\text{orig}}\alpha_i^{\text{orig}}}{\sum_{i=1}^{n}\sigma_{t,i}^{\text{orig}}}. \tag{6}$$

This simple scheme, however, can still lead to limited accuracy. We use Eq. (6) to create all naïve downsampling results.

Table 1: *Definitions of commonly used symbols.*

Symbol	Meaning	Def.
σ_t	Extinction coefficient (density)	§3
σ_s	Scattering coefficient	§3
α	Single-scattering albedo	§3
f	Phase function	§3
$f^{\text{SGGX}}(S)$	SGGX function determined by S	§3
$\hat{f}$	Scaled phase function	§4.1, Eq. (8)
f_j	(Normalized) phase function lobe	§4.1, Eq. (8)
W_j	Lobe weight	§4.1, Eq. (8)
w_j	Lobe weight factor	§6.2, Eq. (12)
$V(i)$	Set of input voxels contained in downsampled voxel i	§5
$V(i,j)$	The j-th cluster of $V(i)$	§5
mean()	Mean pixel intensity	§6
$\text{mean}_k()$	Restricted mean pixel intensity	§6

4 Our Method

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

As demonstrated in prior work [Zhao et al. 2011; Khungurn et al. 2015], material appearance is insensitive to small changes of density σ_t (at a fixed resolution), and the relationship between the two is highly complex. Thus, we focus on computing albedos and phase functions with densities downsampled via Eq. (6).[2]

4.1 Combining Albedo and Phase Function

Because our goal is to compute albedos α and phase functions f at reduced resolutions, it is desirable to have a framework combining these quantities. Given that $\sigma_s(\boldsymbol{\omega}) = \alpha\sigma_t(\boldsymbol{\omega})$, the anisotropic RTE (2) can be rewritten as

$$(\boldsymbol{\omega}\cdot\nabla)L(\boldsymbol{\omega}) = \sigma_t(\boldsymbol{\omega})\left[-L(\boldsymbol{\omega}) + \int_{\mathbb{S}^2} L(\boldsymbol{\omega}')\hat{f}(\boldsymbol{\omega}'\to\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}'\right], \tag{7}$$

where

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

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

$$\hat{f}(\boldsymbol{\omega}'\to\boldsymbol{\omega}) = \sum_{j=1}^{m} W_j f_j(\boldsymbol{\omega}'\to\boldsymbol{\omega}), \tag{8}$$

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

Given a scaled phase function (8), the corresponding effective albedo α^{eff} and phase function f^{eff} can be obtained easily with

$$\alpha^{\text{eff}} = \sum_{j=1}^{m} W_j, \qquad f^{\text{eff}} = \frac{\hat{f}}{\alpha^{\text{eff}}}. \tag{9}$$

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

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

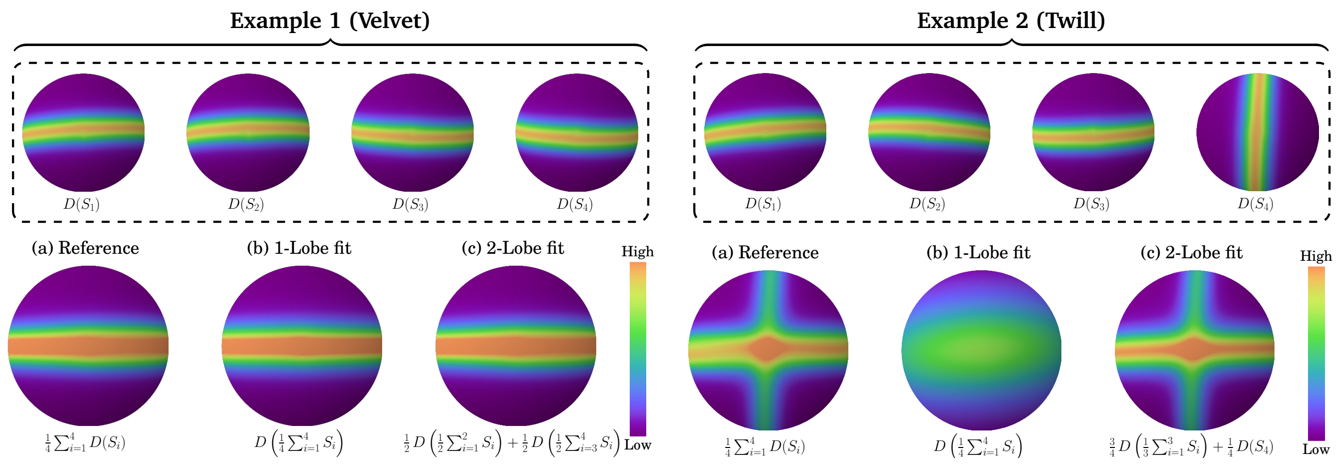

Example 1 (Velvet)
Example 2 (Twill)
$D(S_1)$
$D(S_2)$
$D(S_3)$
$D(S_4)$
(a) Reference
(b) 1-Lobe fit
(c) 2-Lobe fit
High
Low
$\frac{1}{4}\sum_{i=1}^{4} D(S_i)$
$D\left(\frac{1}{4}\sum_{i=1}^{4} S_i\right)$
$\frac{1}{2} D\left(\frac{1}{2}\sum_{i=1}^{2} S_i\right) + \frac{1}{2} D\left(\frac{1}{2}\sum_{i=3}^{4} S_i\right)$
$\frac{3}{4} D\left(\frac{1}{3}\sum_{i=1}^{3} S_i\right) + \frac{1}{4} D(S_4)$

Figure 4: ***Comparison between single- and multi-lobe SGGX fits.*** *In each example, the goal is to fit the reference (a) which equals the average of four SGGX lobes given by S_1, S_2, S_3, and S_4 (visualized in bashed box). Although the single-lobe fit (b) works well in Example 1, it fails to capture the high anisotropy in Example 2. The two-lobe fit (c), in contrast, works much better in this case. See Figure 17 for corresponding rendered images.*

4.2 Input and Output

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

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

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

- Downsampled density $\bar{\sigma}_{t,i}$ given by Eq. (6);
- Scaled phase function $\hat{f}_i$ determined by m SGGX lobes $f_{i,1}, f_{i,2}, \ldots, f_{i,m}$ (which in turn are given by matrices $S_{i,1}, S_{i,2}, \ldots, S_{i,m}$) and m weights $W_{i,1}, W_{i,2}, \ldots, W_{i,m} \in (0,1]$.

4.3 Overview

The rest of this paper focuses on finding a proper scaled phase function $\hat{f}_i$ for each downsampled voxel i. That is, we need to determine the SGGX lobes $f_{i,j}$ (which is equivalent to finding $S_{i,j} \in \mathbb{R}^{3\times3}$) as well as the lobe weights $W_{i,j}$ for $1 \le j \le m$. Since there can be millions of voxels each of which has m unknown matrices $S_{i,j}$ and m unknown weights $W_{i,j}$, the problem is extremely under-constrained in general. To make it tractable, we start with determining $f_{i,j}$ locally based on constituent (input) phase functions (§5), and then optimize the weights $W_{i,j}$ globally (§6).

5 Determining SGGX Lobes

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

The previous phase function downsampling approach [Heitz et al. 2015] focuses on the special case of $m = 1$. In this case, $f_{i,1}$ needs to approximate $\sum_{i' \in V(i)} f_{i'}^{\text{orig}}$ and can be obtained directly via Eq. (5). Namely, $f_{i,1} = f^{\text{SGGX}}(S_{i,1})$ with $S_{i,1} = \left(\sum_{i' \in V(i)} S_{i'}^{\text{orig}}\right)/n$.

Although using a single SGGX function to approximate the average of multiple SGGX functions works adequately in many cases (Figure 4, Example 1), it can lead to limited accuracy for highly anisotropic materials (Figure 4, Example 2).

Phase Function Clustering. To allow $m > 1$, we partition the input phase functions $\{f_{i'}^{\text{orig}} : i' \in V(i)\}$ into m clusters and apply Eq. (5) to each cluster. That is, we set

$$f_{i,j} = f^{\text{SGGX}}\left(\frac{1}{|V(i,j)|} \sum_{i' \in V(i,j)} S_{i'}^{\text{orig}}\right), \tag{10}$$

where $V(i,j) \subset V(i)$ denotes the set containing the indices of all input SGGX phase functions belonging to the j-th cluster. Figure 5 illustrates an example of this process. We compute the clustering $V(i,1), V(i,2), \ldots, V(i,m)$ using K-means with $S_{i'}^{\text{orig}}$ treated as 9D vectors.

To determine the value of m, one can start with $m = 1$ and iteratively increase m until the approximation error, i.e., the L_2 difference between $f_{i,j}$ and $\left(\sum_{i' \in V(i,j)} f_{i'}^{\text{orig}}\right)/n$, stops decreasing rapidly. In practice, we also limit m to 3 for a good balance between model size and result accuracy. See §8.1 for more details.

6 Optimizing Lobe Weights

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

Solving for m weights for each downsampled voxel is still a challenge due to the large number of unknowns. Fortunately, we have a good initial guess to start with: the (linearly) downsampled albedo $\bar{\alpha}_i$ which can be computed using Eq. (6) for each voxel i. Since the scaled phase function is defined as the product of albedo and (normalized) phase function (Eq. (8)), the naïve downsampling approach is equivalent to setting

$$W_{i,j}^{\text{naïve}} = \bar{\alpha}_i |V(i,j)|/|V(i)|, \tag{11}$$

where $|V(i,j)|/|V(i)|$ captures the weight of $f_{i,j}$ due to phase function clustering (see Eq. (10) and Figure 5). In this case, it is easy to verify that $\alpha_i^{\text{eff}} = \sum_{j=1}^{m} W_{i,j}^{\text{naïve}} = \bar{\alpha}_i$.

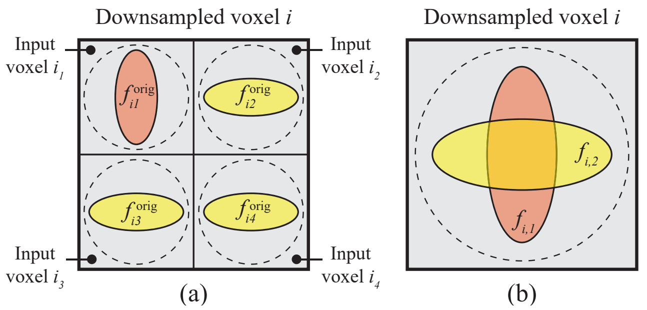


Figure 5: *An example downsampled voxel i containing four input voxels (i.e., $V(i) = \{i_1, i_2, i_3, i_4\}$) is shown in (a), and a fitted two-lobe result in (b). The four input phase functions are partitioned into two clusters $V(i,1) = \{i_1\}$ (in orange) and $V(i,2) = \{i_2, i_3, i_4\}$ (in yellow) based on their shapes. The resulting two-lobe representation is then obtained by applying Eq. (5) to each cluster. Note that this step only determines the shape of each lobe: their weights will be optimized globally in §6.*

6.1 Overview

The rest of this section focuses on finding *weight factors* for each downsampled voxel that can lead to good accuracy. These weight factors scale the lobe weights given by Eq. (11). Because the number of downsampled voxels can be very large (in millions), solving for one set of weight factors per voxel is impractical. Instead, we partition the voxels into K clusters and search for one set of weight factors $\boldsymbol{w}_k := (w_{k,1}, w_{k,2}, \ldots, w_{k,m})$ for each cluster k. Then, for each downsampled voxel i, the lobe weights become

$$W_{i,j} = W^{\text{naïve}}_{i,j}\, w_{c(i),j} = \bar{\alpha}_i \frac{|V(i,j)|}{|V(i)|}\, w_{c(i),j}, \tag{12}$$

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

We describe in §6.2 how to cluster the voxels and reorder the phase function lobes $f_{i,j}$ so that each weight factor $w_{i,j}$ controls a set of lobes with similar shapes. Then, §6.3 introduces our main algorithm that numerically optimizes $\boldsymbol{w}_k$ for each voxel cluster k.

6.2 Voxel Clustering

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

When voxel cluster boundaries go across areas with smoothly varying albedos, the rendered images can occasionally contain visible seams. To ease this problem, we jitter the clusterings by slightly perturbing each $\bar{\alpha}_i$ when performing K-means. Please see §8.1 for examples.

Lobe ordering. Since each weight factor $\boldsymbol{w}_k$ is shared among multiple voxels (i.e., those with $c(i) = k$), the ordering of phase function lobes matters. Intuitively, we need to *reorder* the lobes $f_{i,1}, f_{i,2}, \ldots, f_{i,m}$ for each i so that those with the same indices from different voxels have similar shapes (see Figure 6).

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

$$\pi^*_{i'} = \arg\min_{\pi} \sum_{j=1}^{m} \int_{\mathbb{S}^2} \left[D_{i,j}(\boldsymbol{m}) - D_{i',\pi(j)}(\boldsymbol{m})\right]^2 \mathrm{d}\boldsymbol{m}. \tag{13}$$

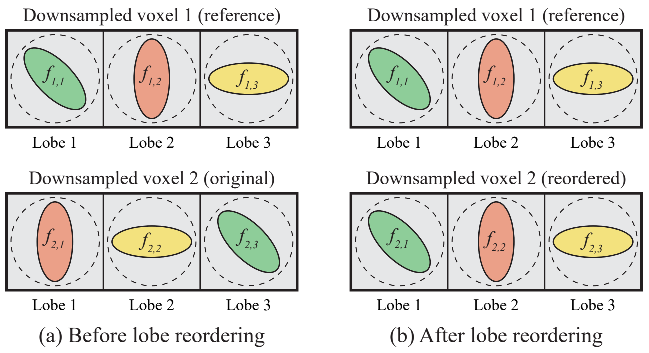


Figure 6: *We **reorder** the phase function lobes so that those with identical indices have similar shapes. In this example, after reordering the lobes of voxel 2 using $\pi^*_2 = (3, 1, 2)$, the shape of $f_{1,j}$ matches that of $f_{2,j}$ for $j = 1, 2, 3$.*

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

6.3 Optimizing Weight Factors

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

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

Training scene setup. To generate the training renderings I and $\tilde{I}$, we need to design lighting and viewing configurations. For ensuring our solved albedo values generalize well to different settings, we perform the fitting using multiple lighting and viewing configurations. In particular, we render the object from the six directions (i.e., X-, X+, Y-, Y+, Z-, Z+) defined by its bounding box (Figure 7). For each view, we render the object using a few spherical harmonic lightings (see §8.1 for more details).

Error metric. To measure the difference between I_r and $\tilde{I}_r$ for each configuration r, we utilize a generalized version of the L_2 distance between mean pixel intensities. Given a rendered image I, let mean(I) denote the average intensity of all (foreground) pixels in I. Previous work [Zhao et al. 2011; Khungurn et al. 2015] uses this measure to define their error metrics as $\sum_r c_r^2 \left(\text{mean}(\tilde{I}_r) - \text{mean}(I_r)\right)^2$ with $c_r = \max(\text{mean}(I_r),\ 0.05)^{-1}$ being a normalization factor. This metric works well when $\tilde{I}_r$ is affected by a small set of spatially invariant parameters, but has difficulties handling more general situations with parameter heterogeneity. In particular, the measure mean($\tilde{I}_r$) only tells if the whole image $\tilde{I}_r$ is too dark or too bright with little information on which voxel clusters are causing the problem, making it difficult to tell which weight factor needs to be corrected. Consequently, the optimization is prone to local minima.

We generalize mean(I) and define the *restricted mean pixel intensity*, denoted as mean$_k$(I), as the average intensity evaluated

[3]Here the smoothness assumption about lighting is a common practice when deriving approximate material properties and is used by the diffusion approximation [Ishimaru 1978] as well as similarity relations [Zhao et al. 2014].

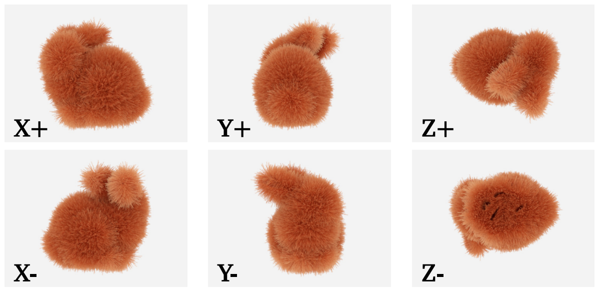


Figure 7: *Six views we use for the training renderings.*

among pixels to which downsampled voxels from cluster k are "directly" visible (see Figure 8). This leads to our error metric:

$$E_{\mathrm{L2}}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) := \sum_r c_r^2 \sum_{k=1}^{K} \left(\mathrm{mean}_k(\tilde{I}_r) - \mathrm{mean}_k(I_r)\right)^2. \quad (14)$$

Our desired weights $\boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_K^*$ minimize Eq. (14). Namely,

$$(\boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_K^*) = \underset{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K}{\mathrm{argmin}}\ E_{\mathrm{L2}}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K). \quad (15)$$

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

Gradient estimation. To solve Eq. (15) using SGD, the key is to obtain an unbiased evaluation of the partial derivative of the error metric (14) with respect to $w_{k,j}$ for any $1 \le k \le K$ and $1 \le j \le m$. It holds that

$$\begin{aligned}\frac{\partial E_{\mathrm{L2}}}{\partial w_{k,j}} &= 2\sum_{r,k'} c_r \left[\mathrm{mean}_{k'}(\tilde{I}_r) - \mathrm{mean}_{k'}(I_r)\right] \frac{\partial \mathrm{mean}_{k'}(\tilde{I}_r)}{w_{k,j}} \\ &= 2\sum_{r,k'} c_r \left[\mathrm{mean}_{k'}(\tilde{I}_r) - \mathrm{mean}_{k'}(I_r)\right] \mathrm{mean}_{k'}\left(\tilde{I}'_{r,k,j}\right),\end{aligned} \quad (16)$$

where $\tilde{I}'_{r,k,j}$ is the gradient image with respect to $w_{k,j}$ in which each pixel p has intensity $\tilde{I}'_{r,k,j}(p) = (\partial \tilde{I}_r / \partial w_{k,j})(p)$. We use path tracing to obtain an unbiased evaluation of $\tilde{I}'_{r,k,j}$. Please refer to Appendix A for more details. Figure 9 shows an example of original and gradient images.

Stochastic gradient descent. With the gradient values, we can apply stochastic gradient descent (SGD) to find weight factors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ minimizing Eq. (14). We use $\boldsymbol{w}_k^{(0)} = (1, \ldots, 1)$ for all $k = 1, 2, \ldots, K$ as the initial guesses. In each iteration, we

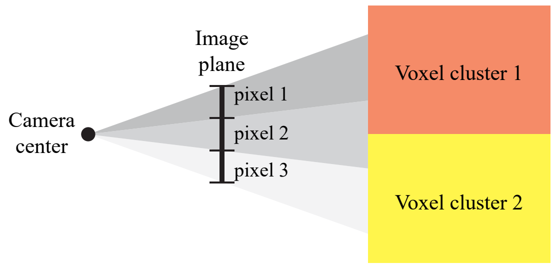


Figure 8: *We define* **restricted mean pixel intensity** *that separates errors caused by different voxel clusters and yields better convergence of the optimization. In this 2D example has three pixels and two voxel clusters, mean$_1$() is computed over pixels 1 and 2 because (at least part of) voxel cluster 1 is visible through these two pixels. Similarly, mean$_2$() involves pixels 2 and 3.*

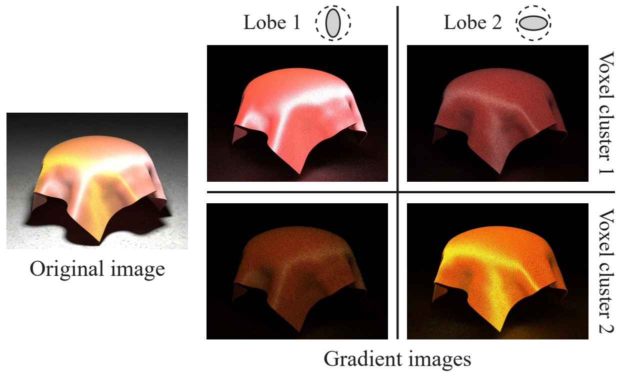


Figure 9: *Original and gradient images rendered with our approach. This fabric contains differently colored warp (in pink) and weft (in yellow) yarns. Our method uses two voxel clusters (i.e., one for each type of yarn) and two phase function lobes per voxel. The corresponding gradient images successfully capture the main color of each cluster as well as anisotropy of the lobes. Furthermore, the varying average intensities of the gradient images demonstrate the correlation between voxel clusters and lobe weights: voxels in cluster 1 have high weights for lobe 1, and vice versa.*

update each weight factor $w_{k,j}$ via:

$$w_{k,j}^{(t+1)} = w_{k,j}^{(t)} - \delta_t \left.\frac{\partial E_{\mathrm{L2}}}{\partial w_{k,j}}\right|_{w_{k,j}^{(t)}}, \quad (17)$$

where δ_t is the step size of iteration t. The step size gradually decreases during iterations. In theory, δ_t needs to satisfy $\sum_{t=1}^{\infty} \delta_t = \infty$ and $\sum_{t=1}^{\infty} \delta_t^2 < \infty$ for ensuring convergence. We follow the common practice [Khungurn et al. 2015] of using the divergent harmonic series by setting

$$\delta_t = a_{k,j}/(t + b_{k,j}), \quad (18)$$

where $a_{k,j}$ and $b_{k,j}$ are constants. How to obtain the values for these constants is discussed in §8.1.

6.4 Handling Multiple Color Channels

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

In practice, however, we found that the input parameters are usually *semi-colored*: only albedos vary between color channels while densities and phase functions stay constant. In this case, we keep SGGX lobes $f_{i,j}$ and weight factors $w_{i,j}$ single-channel and introduce an extra colored scaling factor $\boldsymbol{s}_k := (s_k^R, s_k^G, s_k^B)$ for each voxel cluster k, so that the lobe weights $W_{i,j}$ originally defined in Eq. (12) become colored with

$$W_{i,j}^{\mathrm{clr}} = s_{c(i)}\, \bar{\alpha}_i^{\mathrm{clr}} \frac{|V(i,j)|}{|V(i)|} w_{c(i),j} \quad \text{for } \mathrm{clr} \in \{R, G, B\}, \quad (19)$$

where $c(i)$ denotes the index of the cluster to which voxel i belongs, and $\bar{\alpha}_i^{\mathrm{clr}}$ is the corresponding (i.e., red, green, or blue) component of the downsampled albedo at voxel i. We optimize $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K$ together with the weight factors $\boldsymbol{w}_{i,j}$ using SGD as this is more efficient than optimizing $\boldsymbol{w}_{i,j}$ for each color-channel separately (see §8.1 for an example).

6.5 Discussion

When $m = 1$ and $K = 1$, the scaled phase function at each voxel i becomes $\hat{f}_i = \bar{\alpha}_i\, w\, f_{i,1}$, and the optimization reduces to searching

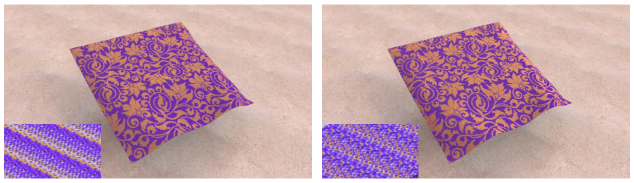


Figure 10: *Two* $(16\times)^3$*-downsampled results synthesized with the same design but differently stacked exemplar volumes (whose tiled versions are shown as insets). The two exemplar stacking schemes have lead to visually identical results.*

for a global weight factor w which effectively scales all albedos uniformly. Furthermore, if the input model has homogeneous albedo α, namely $\bar{\alpha}_i \equiv \alpha$ for all i, the search for w becomes equivalent to finding an altered albedo $\tilde{\alpha} := \alpha w$. This is precisely what previous methods such as [Zhao et al. 2011] and [Khungurn et al. 2015] do. Our method, on the other hand, is much more general and allows multiple lobes and voxel clusters as well as heterogeneous input albedos.

Voxel clustering. Our voxel clustering (§6.2) relies purely on downsampled albedo $\bar{\alpha}_i$ for each voxel i. This simple scheme has yielded high-quality results for all our experiments (see §8). In the future, more sophisticated features capturing spatial locations and/or phase functions can be used for handling input with higher degrees of correlations.

7 Exploiting Modularity

Many micro-appearance models consist of multiple *blocks* each of which comes form a predefined set of *exemplars*. For instance, Zhao et al. [2012] introduced a *structure-aware synthesis* algorithm that automatically constructs highly complex fabric models based on exemplars with elementary weave patterns.

We exploit such modularity to accelerate the optimization of lobe weights. Our high-level idea is to directly downsample the exemplars (using techniques introduced in §5 and §6) as precomputation. Then, to obtain a synthesized model at lower-resolution, we can directly replace its blocks with our downsampled versions. Because a synthesized model is normally much larger (in terms of the number of blocks) than the corresponding exemplars, it is more efficient to downsample the latter. More importantly, this allows many models synthesized from one set of exemplars to share one precomputation, greatly reducing the amortized downsampling overhead.

Downsampling exemplars. We downsample all exemplars together by stacking them (randomly) into a single *exemplar volume* and downsampling it using techniques introduced in §5 and §6. Doing so has the following benefits. First, it guarantees downsampled voxels from different exemplars to have similar scaled phase functions if they contain similar input albedo and phase functions. This consistency is important for avoiding visible discontinuities in synthesized results. Second, it offers better performance than optimizing each exemplar separately.

In theory, because of high-order multiple scattering across neighboring exemplars, different ways of stacking can lead to varying optimization results. Fortunately, as demonstrated in Figure 10, we found that this hardly happens in practice: downsampled parameters using different stacking schemes generally yield visually identical results. Therefore, we compose the exemplar volume by stacking together individual exemplars in a randomized manner.

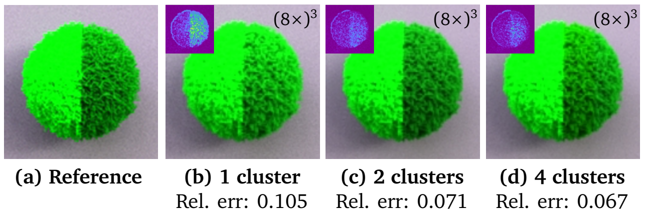


Figure 11: ***Determining the number of voxel clusters.*** *A hairy ball made with two materials (a). Each of these materials requires a distinctive albedo scaling to match the ground truth. If using only one cluster, the two materials are forced to be treated equally, resulting in poor accuracy (b). Using two clusters with our clustering scheme, each of the two materials will be handled separately, offering better accuracy (c). Going beyond two clusters (d), on the other hand, has limited benefits.*

8 Results

Using our technique described in Sections 5, 6 and 7, scaled phase functions (or, equivalently, albedos and phase functions) can be computed at greatly reduced resolutions. This section shows results generated using our method. In §8.1, we empirically evaluate and justify several components of our approach. Then, in §8.2 we show downsampled results for a range of objects represented with high-resolution anisotropic volumes.

8.1 Evaluations and Justifications

Voxel clustering. As described in §6.2, our method groups downsampled voxels into K clusters to make the optimization of weight factors tractable. In our experience, using one to five clusters generally provides a good balance between accuracy and performance (see Figure 11). We also jitter the voxel clustering for input with continuously changing albedos to ease potential seams (Figure 12).

Lighting in training renderings. We use four SH lightings (i.e., first two bands) for our training renderings. Figure 13 shows an example where we optimized the homogeneous twill (Figure 3) using one, four, and nine SH lightings (corresponding to band-zero, one, and two) and the six views shown in Figure 7. The L_2 errors are computed between the reference image and ones rendered using parameters obtained after a certain number of SGD iterations. The results indicate that using more than four SH bases yields little benefit in terms of accuracy. Thus, we use four SH lightings for our training renderings.

Harsh lighting. We use SH lighting for the training renderings to allow the optimized parameters to generalize well to arbitrary smooth lighting conditions. In theory, when these parameters are rendered under harsh lighting, the resulting accuracy can degrade. However, we did not observe such cases in practice:

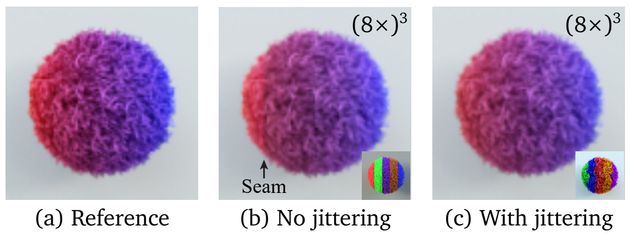


Figure 12: ***Avoiding visible seams*** *by jittering voxel clustering. When the input has smoothly changing albedos (a), neighboring voxel clusters can have clear boundaries that may yield visible seams (b). To ease this problem, we slightly jitter the clusters so that their boundaries become fuzzier (c).*

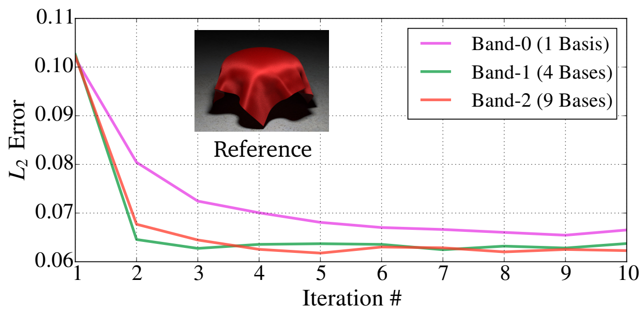


Figure 13: ***Selecting training lighting.*** *The errors are evaluated between a reference image and those rendered using optimized parameters after a certain number of SGD iterations. An area light source is used to generate all these renderings. The results indicate that going beyond four SH bases offers little improvement for accuracy. The small differences between L_2 errors for band-1 and band-2 are mostly due to Monte Carlo noise.*

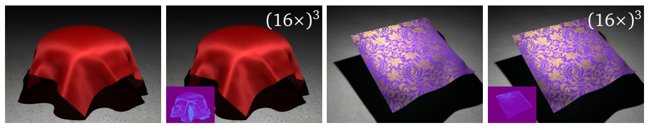
our optimized parameters generalize well even to extremely high-frequency lightings (see Figure 14 for two examples).

Handling multiple color channels. As described in §6.4, we optimize one extra term $\mathbf{s}_k$ for each voxel cluster k to handle *semi-colored* input where only single-scattering albedo varies between different color channels. As shown in Figure 15, this is more efficient than optimizing $w_{i,j}$ for each color-channel separately while providing similar accuracy.

Number of phase function lobes. As discussed in §5, we determine the number of phase function lobes m by iteratively increasing m until the approximation error stops descending rapidly. In particular, we stop increasing m when the L_2 error with $(m+1)$ lobes is greater than 50% of that with m lobes. Figure 16 shows how the approximation error changes with the number of lobes for the example from Figure 4. Based on the aforementioned scheme, we use two lobes for the twill while one for the velvet. Figure 17 contains the corresponding renderings justifying our choices of lobe counts. Further, we limit m to 3 in all our experiments for a good balance between model size and result accuracy. Lastly, notice that when using only one lobe (i.e., $m = 1$), our method effectively optimizes only the downsampled albedos $\tilde{\alpha}_i$.

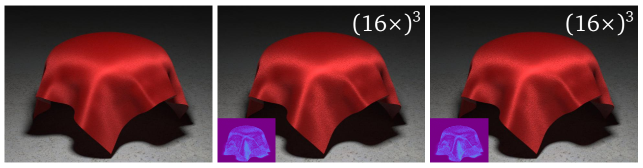
8.2 Main Results

We evaluate the effectiveness of our approach using six examples. We tune $a_{k,j}$ and $b_{k,j}$ in Eq. (18) manually using a subset of lighting/viewing conditions so that the error metric (14) changes neither too quickly (resulting in oscillation) nor too slowly (leading to slow convergence). The overhead for this extra tuning is less than 10% of the total optimization time. Since we use low-resolution (around 100p) and noisy training renderings, the total optimization time for each object is comparable to the rendering time for one image at 720p. Our optimization configurations and performance numbers are summarized in Table 2. The values of all step sizes $a_{k,j}$, $b_{k,j}$ as well as optimized weight factors $w_{k,j}$ and



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

Figure 14: *Although our training renderings use smooth SH lightings, the optimized parameters generalize well to harsh lighting conditions. These two examples use local point light sources which lead to sharp shadow boundaries on the ground.*



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

Figure 15: *For semi-colored input, optimizing multiple color channels jointly (b) is more efficient than optimizing each channel independently (c) while yielding similar resulting accuracy.*

colored scaling factors $\mathbf{s}_k$ are available as supplemental material. We use a modified version of the Mitsuba renderer [Jakob 2013] to generate all regular and gradient images.

Figure 18 shows lower-resolution volumes generated with our approach. The first three examples have homogeneous albedos while the last two have heterogeneous ones. Each object is rendered under two environmental lightings that are significantly more complicated than our training ones (i.e., four SH lightings).

The first row of Figure 18 contains a hairy bunny. After downsampling at $(4\times)^3$, our optimized parameters using one lobe per voxel generalize well to different environmental lightings. The second row of Figure 18 shows a shiny twill fabric originally represented with 4.76×10^{11} effective voxels. Because of the highly anisotropic nature of this material, we use two lobes per voxel. Our result, which contains only 1.07×10^8 effective voxels, achieves more than three orders of magnitude storage saving and successfully preserves this material's glossy appearance under both local (left) and global (right) lighting. The third row of Figure 18 contains a highly scattering velvet with 7.77×10^{11} effective voxels. The velvet's highly detailed surface structure caused naïve downsampling to have very poor accuracy. Our model computed with one lobe per pixel, on the other hand, has 2.03×10^7 effective voxels and maintains good accuracy while providing four orders of magnitude data reduction. The quality of our optimized parameters for this model is further demonstrated in the supplemental video. In the fourth row, we show a hairy ball with gradually changing colors. Our optimization uses five voxel clusters with jittering (visualized in the insets) and one lobe per voxel. The result achieves good accuracy while reducing the storage by almost three orders of magnitude. The bottom row of Figure 18 contains another complex twill scene with differently colored warps and wefts (identical to Figure 9). This heterogeneity results in distinctive dual-colored anisotropic highlights. Our result based on two voxel clusters and two-lobe phase functions maintains the appearance of this object well even at the yarn-level. Please refer to the supplemental video for an animated result of this scene.

Figure 19 shows downsampled results from two damask fabrics synthesized from a single set of 120 example blocks. We downsample the exemplars using three voxel clusters and three lobes per voxel. This precomputation takes 60 core hours. Then, the two downsampled damasks are obtained by stacking the pre-downsampled example blocks. Based on the single precompu-

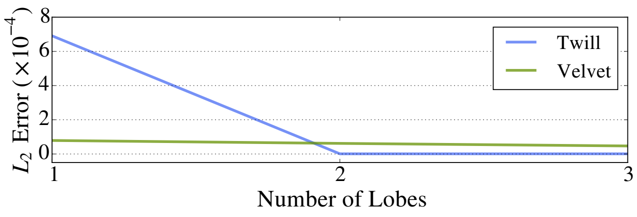


Figure 16: *Approximation error (measured in L_2) decreases as the number of phase function lobes increases.*

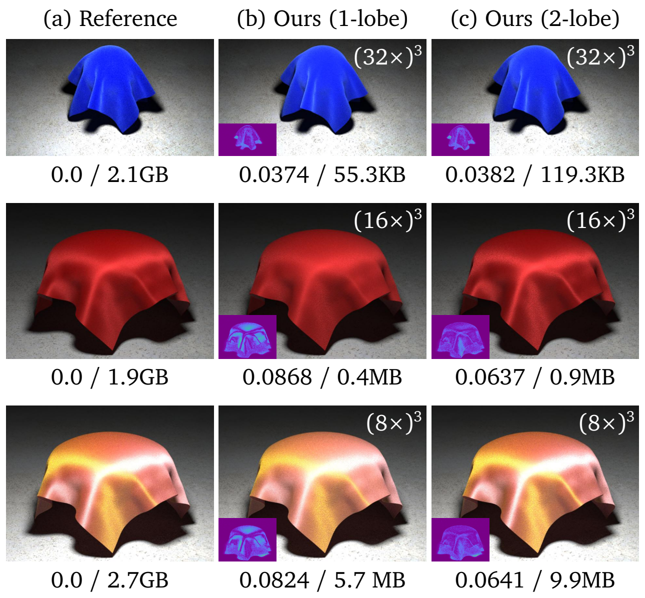


Figure 17: *Optimized results using* ***varying numbers of phase function lobes****. Relative error and data size are shown below each image. For each example, optimizations corresponding to (b) and (c) take similar time to converge. For the velvet (the top row), one lobe suffices. For the twill (the bottom two rows) with high anisotropy, on the other hand, using two lobes has led to superior accuracy.*

Table 2: *Stochastic gradient descent settings (i.e., number of voxel clusters K and number of phase function lobes per voxel m) and optimization time (in CPU core hours) for all results in Figures 18 and 19. Our optimization time is comparable to the rendering time (shown in the last column) for one image at 720p.*

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

tation, our approach results in good accuracy while reducing the amount of data by three orders of magnitude.

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

9 Conclusion

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

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions. This work was supported by the National Science Foundation (IIS 1451828), the UC San Diego Center for Visual Computing, and AWS Cloud Credits for Research.

References

Bruneton, E., and Neyret, F. 2012. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics 18*, 2, 242–260.

Chandrasekhar, S. 1960. *Radiative Transfer*. Dover Publications Inc.

Cook, R. L., Halstead, J., Planck, M., and Ryu, D. 2007. Stochastic simplification of aggregate detail. *ACM Trans. Graph. 26*, 3, 79:1–79:8.

Gkioulekas, I., Zhao, S., Bala, K., Zickler, T., and Levin, A. 2013. Inverse volume rendering with material dictionaries. *ACM Trans. Graph. 32*, 6, 162:1–162:13.

Han, C., Sun, B., Ramamoorthi, R., and Grinspun, E. 2007. Frequency domain normal map filtering. *ACM Trans. Graph. 26*, 3, 28:1–28:12.

Hašan, M., and Ramamoorthi, R. 2013. Interactive albedo editing in path-traced volumetric materials. *ACM Trans. Graph. 32*, 2, 11:1–11:11.

Heitz, E., Dupuy, J., Crassin, C., and Dachsbacher, C. 2015. The SGGX microflake distribution. *ACM Trans. Graph. 34*, 4, 48:1–48:11.

Ishimaru, A. 1978. *Wave propagation and scattering in random media*, vol. 2. Academic press New York.

Jakob, W., Arbree, A., Moon, J. T., Bala, K., and Marschner, S. 2010. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph. 29*, 4, 53:1–53:13.

Jakob, W., Hašan, M., Yan, L.-Q., Lawrence, J., Ramamoorthi, R., and Marschner, S. 2014. Discrete stochastic microfacet models. *ACM Trans. Graph. 33*, 4, 115:1–115:10.

Jakob, W., 2013. Mitsuba physically-based renderer. http://www.mitsuba-renderer.org.

Jarabo, A., Wu, H., Dorsey, J., Rushmeier, H., and Gutierrez, D. 2014. Effects of approximate filtering on the appearance of bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics 20*, 6, 880–892.

Khungurn, P., Schroeder, D., Zhao, S., Bala, K., and Marschner, S. 2015. Matching real fabrics with micro-appearance models. *ACM Trans. Graph. 35*, 1, 1:1–1:26.

Kraus, M., and Bürger, K. 2008. Interpolating and downsampling RGBA volume data. In *VMV*, 323–332.

Liu, X., Tanaka, M., and Okutomi, M. 2012. Noise level estimation using weak textured patches of a single noisy image. In *IEEE International Conference on Image Processing (ICIP) 2012*, 665–668.

Meng, J., Papas, M., Habel, R., Dachsbacher, C., Marschner, S., Gross, M., and Jarosz, W. 2015. Multi-scale modeling and rendering of granular materials. *ACM Trans. Graph. 34*, 4, 49:1–49:13.

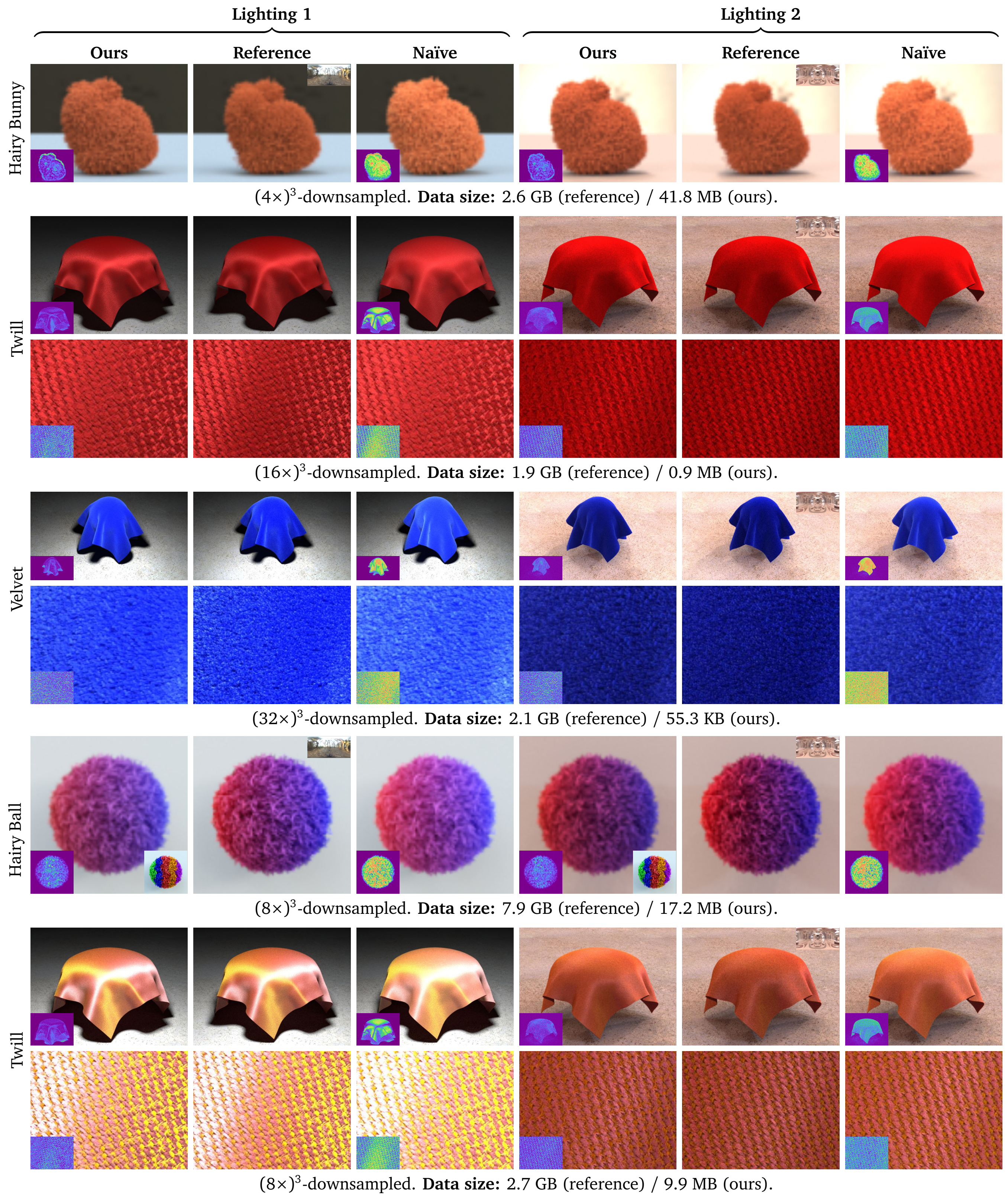

Lighting 1
Lighting 2
Ours
Reference
Naïve
Ours
Reference
Naïve
Hairy Bunny
$(4\times)^3$-downsampled. **Data size:** 2.6 GB (reference) / 41.8 MB (ours).
Twill
$(16\times)^3$-downsampled. **Data size:** 1.9 GB (reference) / 0.9 MB (ours).
Velvet
$(32\times)^3$-downsampled. **Data size:** 2.1 GB (reference) / 55.3 KB (ours).
Hairy Ball
$(8\times)^3$-downsampled. **Data size:** 7.9 GB (reference) / 17.2 MB (ours).
Twill
$(8\times)^3$-downsampled. **Data size:** 2.7 GB (reference) / 9.9 MB (ours).

Figure 18: Main results. *Our reduced-resolution representations provide good accuracy under general lighting with up to four orders of magnitude storage saving. Environmental lightings used in these renderings are visualized in the top-right corner of individual reference images. Those without lighting visualizations use local area light sources. Please refer to Table 2 for optimization and rendering times.*

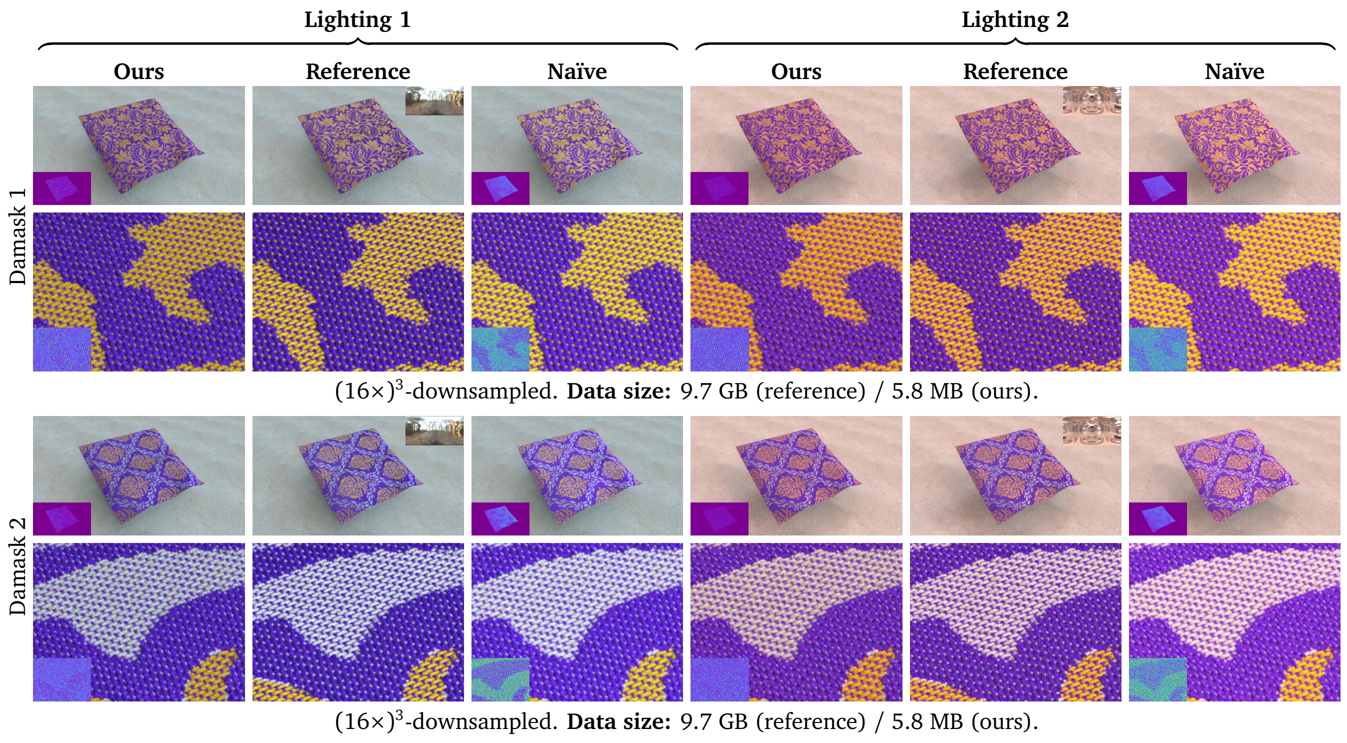

Lighting 1
Lighting 2
Ours
Reference
Naïve
Ours
Reference
Naïve
Damask 1
$(16\times)^3$-downsampled. **Data size:** 9.7 GB (reference) / 5.8 MB (ours).
Damask 2
$(16\times)^3$-downsampled. **Data size:** 9.7 GB (reference) / 5.8 MB (ours).

Figure 19: Exploiting modularity. *The two downsampled damask fabrics sharing one optimization offer good accuracy and three orders of magnitude storage saving. The corresponding environmental lightings are shown in top-right corners of the reference images. Please refer to Table 2 for optimization and rendering times.*

SICAT, R., KRUGER, J., MOLLER, T., AND HADWIGER, M. 2014. Sparse PDF volumes for consistent multi-resolution volume rendering. *IEEE Transactions on Visualization and Computer Graphics 20*, 12, 2417–2426.

TAN, P., LIN, S., QUAN, L., GUO, B., AND SHUM, H.-Y. 2005. Multiresolution reflectance filtering. In *Proceedings of the 16th Eurographics Conference on Rendering Techniques*, 111–116.

WESTIN, S. H., ARVO, J. R., AND TORRANCE, K. E. 1992. Predicting reflectance functions from complex surfaces. *SIGGRAPH Comput. Graph. 26*, 2, 255–264.

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2011. Building volumetric appearance models of fabric using micro CT imaging. *ACM Trans. Graph. 30*, 4, 44:1–44:10.

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2012. Structure-aware synthesis for predictive woven fabric appearance. *ACM Trans. Graph. 31*, 4, 75:1–75:10.

ZHAO, S., RAMAMOORTHI, R., AND BALA, K. 2014. High-order similarity relations in radiative transfer. *ACM Trans. Graph. 33*, 4, 104:1–104:12.

A Rendering Gradient Images

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

$$\tilde{I}_r(p) = \int_\Omega f(\bar{\boldsymbol{x}})\,\mathrm{d}\bar{\boldsymbol{x}}, \tag{20}$$

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

$$f(\bar{\boldsymbol{x}}) = \left(\prod_{v=1}^{n} T_v\,\bar{\sigma}_{t,i_v}\,F_v\right) L_e(\boldsymbol{x}_{n+1}, \boldsymbol{x}_n), \tag{21}$$

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

$$F_v := \hat{f}_{i_v}\left(\frac{\boldsymbol{x}_v - \boldsymbol{x}_{v-1}}{\|\boldsymbol{x}_v - \boldsymbol{x}_{v-1}\|} \to \frac{\boldsymbol{x}_{v+1} - \boldsymbol{x}_v}{\|\boldsymbol{x}_{v+1} - \boldsymbol{x}_v\|}\right). \tag{22}$$

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

$$\tilde{I}_r'(p) := \frac{\partial}{\partial w}\tilde{I}_r(p) = \int_\Omega \frac{\partial f}{\partial w}(\bar{\boldsymbol{x}})\,\mathrm{d}\bar{\boldsymbol{x}}. \tag{23}$$

The problem then boils down to differentiating $f(\bar{x})$ with respect to w. Assume without loss of generality that w affects the scaled phase functions F_v for $1 \le v \le n_0$. Then $f(\bar{\boldsymbol{x}}) = g(\bar{\boldsymbol{x}}) \prod_{v=1}^{n_0} F_v$, where $g(\bar{\boldsymbol{x}})$ captures all terms in Eq. (21) that do not depend on w. It follows that

$$\frac{\partial f}{\partial w}(\bar{\boldsymbol{x}}) = g(\bar{\boldsymbol{x}}) \frac{\partial}{\partial w} \prod_{v=1}^{n_0} F_v = g(\bar{\boldsymbol{x}}) \sum_{v=1}^{n_0} \left[\frac{\partial}{\partial w} F_v \prod_{v' \ne v} F_{v'}\right], \tag{24}$$

where $\frac{\partial}{\partial w} F_v$ can be obtained via Eq. (22), Eq. (8) and Eq. (12). In practice, we compute Eq. (23) using unidirectional path tracing.