Interactive Albedo Editing in Path-Traced Volumetric Materials

MILOŠ HAŠAN and RAVI RAMAMOORTHI
University of California, Berkeley

Materials such as clothing or carpets, or complex assemblies of small leaves, flower petals, or mosses, do not fit well into either BRDF or BSSRDF models. Their appearance is a complex combination of reflection, transmission, scattering, shadowing, and inter-reflection. This complexity can be handled by simulating the full volumetric light transport within these materials by Monte Carlo algorithms, but there is no easy way to construct the necessary distributions of local material properties that would lead to the desired global appearance. In this article, we consider one way to alleviate the problem: an editing algorithm that enables a material designer to set the local (single-scattering) albedo coefficients interactively, and see an immediate update of the emergent appearance in the image. This is a difficult problem, since the function from materials to pixel values is neither linear nor low-order polynomial. We combine the following two ideas to achieve high-dimensional heterogeneous edits: precomputing the homogeneous mapping of albedo to intensity, and a large Jacobian matrix, which encodes the derivatives of each image pixel with respect to each albedo coefficient. Combining these two datasets leads to an interactive editing algorithm with a very good visual match to a fully path-traced ground truth.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Global Illumination*

General Terms: Algorithms

Additional Key Words and Phrases: Material editing, volume scattering, radiative transfer, first-order Taylor approximations

ACM Reference Format:

Hašan, M. and Ramamoorthi, R. 2013. Interactive albedo editing in path-traced volumetric materials. ACM Trans. Graph. 32, 2, Article 11 (April 2013), 11 pages.
DOI: http://dx.doi.org/10.1145/2451236.2451237

Major funding for this work was provided by NSF grant 1011832, "Beyond Flat Images: Acquiring, Processing and Fabricating Visually Rich Material Appearance". Additional support was provided by NSF grant 1115242, ONR PECASE grant N00014-09-1-0741, Okawa Foundation Research Grant, the Intel Science and Technology Center for Visual Computing, and funding from NVIDIA, Adobe, and Pixar.

Authors' addresses: M. Hašan (corresponding author) and R. Ramamoorthi, University of California, Berkeley, CA; email: milos.hasan@gmail.com.

DOI: http://dx.doi.org/10.1145/2451236.2451237

1. INTRODUCTION

Materials consisting of intricately intertwined fibers, such as clothing or carpets, or organic materials such as complex assemblies of small leaves, flower petals, or mosses, do not fit well into either BRDF or BSSRDF models. While they exhibit significant amounts of internal light scattering, they are not locally well approximated by flat slabs, for which this scattering could be solved analytically. Instead, their appearance is a complex combination of reflection, transmission, scattering, shadowing, and inter-reflection. This complexity can, however, be handled by simulating the full volumetric light transport within these materials by Monte Carlo algorithms.

Rendering of cloth at the yarn or fiber level has recently received significant attention; for example, Jakob et al. [2010] introduced an anisotropic formulation of radiative transfer especially suitable for cloth. Zhao et al. [2011] have shown full light transport simulations of highly detailed volumetric models of cloth, constructing the fiber distributions from micro-CT scans of cloth samples. Schröder et al. [2011] model fibers as hairs, and introduce a rendering method that uses the statistical spatial distribution of fibers rather than their precise geometry. We believe that similar Monte Carlo simulations, which have been considered intractable in the past, are going to become feasible within a few years for a much wider range of materials.

However, one problem that prevents wider adoption of this approach is that there is no easy way to construct the necessary distributions of local material properties that would lead to the desired global appearance. In this article, we consider one way to alleviate the problem: an editing algorithm to help a material designer set the local *single-scattering albedo* coefficients interactively, and see an immediate update of the emergent appearance in the image. We focus on optically thick materials; while some of our derivations also hold for thin materials like fog, our approximations may not be so good, and we have not explored these materials in depth.

Figure 1 shows an example edit using our approach. We start from a homogeneous distribution and add high-resolution variation by modifying the heterogeneous single-scattering albedo coefficients, and finally adjust the overall mean free path of the material. Our algorithm computes the update in material appearance at interactive rates, including updates in the light diffusion, shadowing, and inter-reflection. Figure 2 illustrates these emergent effects by separating direct and indirect illumination components.

To achieve these edits, we study the function that maps the values of material coefficients into pixel values, assuming a static scene with fixed lighting. If we keep the properties of the scarf homogeneous, we can solve the albedo editing problem by precomputing the albedo-intensity mapping: a one-dimensional, increasing function for every pixel that records the dependence of intensity I on the single-scattering albedo α. On the other hand, if the number of albedo coefficients is high (in the thousands or millions), this becomes a high-dimensional, nonlinear function; no general methods to compress such functions into simpler ones are known. However, the function clearly has special properties: increasing the scattering albedo of one small yarn segment of a cloth model will increase the intensity of a few pixels where the segment projects, while nearby

Fig. 1. Left: A scarf modeled volumetrically as a fiber density distribution, rendered using full Monte Carlo light transport with multiple scattering. Middle: Using our algorithm, the user can place heterogeneous patterns onto the scarf by independently setting the single-scattering albedos of over 260,000 yarn segments and observing an interactive update of the light transport in the image. Right: An extension of our method allows a change in the mean free path of the material, which makes the scarf appear more fluffy. (Scarf model from Kaldor et al. [2008], use approved by Steve Marschner.)

pixels will receive a subtle, low-frequency update. This suggests similarity to a large body of previous work on precomputed light transport, where similar edits are achieved by precomputing a light transport matrix.

In this article, we combine these two ideas to achieve high-dimensional heterogeneous edits: precomputing the albedo-intensity curve and a large Jacobian matrix, which encodes the derivatives of each image pixel with respect to each albedo coefficient. We also show extensions of the technique to combining multiple Jacobians, and to editing the overall mean free path of the material.

2. RELATED WORK

Relighting and precomputed light transport. Our work is related to precomputed methods based on the linearity of light transport [Nimeroff et al. 1994; Dorsey et al. 1995; Sloan et al. 2002; Ng et al. 2003], but we focus on material rather than light editing. We refer the reader to Ramamoorthi [2009] for a survey of recent work in the area of precomputed light transport methods. Our goal is to extend this rich (but linear) theory to the nonlinear problem of editing heterogeneous material albedos. Recently, a precomputation approach has been applied to volumetric materials by Bouthors et al. [2008], but not with the purpose of editing the materials.

BRDF editing. Material editing in the context of a Whitted ray-tracer has been explored by Séquin and Smyrl [1989]. Ben-Artzi et al. [2006] have explored BRDF editing under direct illumination from an environment map. A solution for global illumination effects has been included into a BRDF editing framework by Ben-Artzi et al. [2008] and Sun et al. [2007]. The key observation in these papers is that editing n BRDF multipliers in a scene with d light bounces leads to an n-variable polynomial of degree d. Unfortunately, representing this explicitly is only practical for 1 or 2 bounces; higher bounces can be approximated by a single ambient term, but this is insufficient for scenes with significant multiple bounces or scattering. In contrast, our approach uses a precomputed nonlinear curve for homogeneous edits, combined with a precomputed first-order approximation of heterogeneity; both of these components are practically tractable in terms of storage, even for very long light paths (we use light paths of length 100 in all our examples).

Subsurface scattering editing. The SubEdit paper [Song et al. 2009] uses a factored representation for editing the heterogeneous BSSRDF of a translucent object. Wang et al. [2008] use the diffusion equation to model subsurface scattering, and present a GPU implementation fast enough to be used for real-time editing of the material properties. However, these techniques (or any other approaches based on BSSRDFs and/or diffusion approximations) are not easily applicable to materials that are not locally well

Fig. 2. A separation of the direct (single-scattering) and indirect (multiple-scattering) components of our scenes, illustrating their volumetric nature and the importance of both short and long light paths. (Dragon model from Stanford 3D scanning repository.)

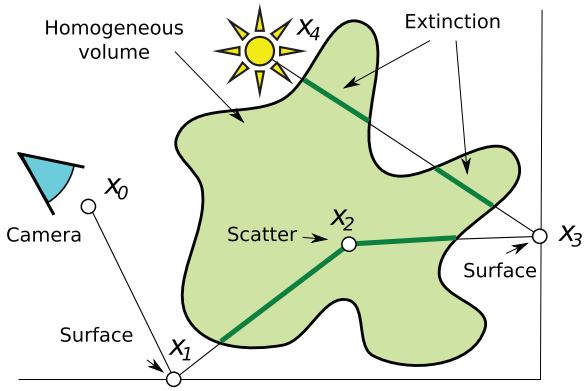

Fig. 3. Our mathematical framework is based on the path integral formulation of light transport [Veach 1997], extended to account for volumetric effects [Pauly et al. 2000]. This figure shows an example light path that interacts with both surfaces and volumes. For simplicity, a homogeneous volume is shown.

approximated by flat slabs, for example, for our scarf and rose examples. Furthermore, these methods do not include global illumination.

3. MATHEMATICAL FRAMEWORK

Previous precomputed light transport methods derive either from the direct lighting equation for environment maps, or the rendering equation of Kajiya [1986]. The material editing framework of Ben-Artzi et al. [2008] derives from the operator formulation of Arvo [1993]. However, in our case the path formulation of global illumination by Veach [1997] is much more natural, because it can account for any combination of surface and volume scattering events within one consistent framework. Since it expresses pixel intensity as a single integral, it also greatly facilitates the computation of derivatives with respect to material coefficients.

3.1 Path Formulation of Light Transport for Surfaces

In Veach's formulation, the intensity of a pixel i is expressed as an integral over all light paths in the scene passing through this pixel. The contribution of each path is a product of a number of terms, some of which depend on the materials in the scene.

$$I_i = \int_\Omega f_i(\bar{x}) d\mu(\bar{x}) \tag{1}$$

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, $\bar{x} = x_0 x_1 \ldots x_l$, such that x_0 is the camera position, x_1 is the surface point directly visible through the pixel, x_l is on a light source, and $x_2, \ldots, x_{l-1}$ are any light bounce points in the scene.

The contribution $f_i(\bar{x})$ of any single path $\bar{x} = x_0 x_1 \ldots x_l$ is the product of a pixel weight *term* $W_i(x_0 \leftarrow x_1)$, a light emission term $L_e(x_{l-1} \leftarrow x_l)$, geometry terms $G(x_k \leftrightarrow x_{k+1}, z)$ corresponding to each segment of the path, and material terms $M(x_{k-1} \leftarrow x_k \leftarrow x_{k+1}, z)$ corresponding to every vertex.

$$f_i(\bar{x}) = W_i(x_0 \leftarrow x_1) \cdot L_e(x_{l-1} \leftarrow x_l) \prod_{k=1}^{l-1} G(x_k \leftrightarrow x_{k+1}) M(x_{k-1} \leftarrow x_k \leftarrow x_{k+1})$$

As usual, we denote by arrows the direction of light flow, and use a two-way arrow to indicate symmetry. Let's look at these terms in more detail.

—The *pixel weight* $W_i(x_0 \leftarrow x_1)$ expresses how much of the path contributes to the pixel. If we assume a box filter and a pinhole camera model, the pixel weight becomes just a binary indicator function that equals 1 if the path passes through pixel i and 0 otherwise. Extensions to other filters and camera models are possible, and orthogonal to our editing technique.

—The *emission term* $L_e(x_{l-1} \leftarrow x_l)$ is the radiance emitted by the surface point x_l in the direction of x_{l-1}. Strictly speaking, this formulation only works for area light sources, but extensions to point lights and environment maps are again possible and orthogonal to material editing.

—The *geometry term* $G(x_k \leftrightarrow x_{k+1})$ is a product of the binary visibility function $V(x_k, x_{k+1})$ that is 0 if the given segment contains occlusion and 1 otherwise, the $1/\|x_i - x_{i+1}\|^2$ distance fall-off term (except the first camera segment and segments connecting to infinitely distant lights), and cosine terms between the segment and the local normal on each surface or area light vertex.

—The *material term* $M(x_{k-1} \leftarrow x_k \leftarrow x_{k+1})$ is normally equal to the BRDF $f_r(x_{k-1} \leftarrow x_k \leftarrow x_{k+1}, z)$. It can also be extended to two-sided BSDFs (e.g., the rough glass model of Walter et al. [2007]).

3.2 Extending the Path Formulation to Volumes

Veach's formulation can be extended to handle volumetric scattering and absorption as well: the vertices x_k can be created inside or on the boundaries of volumetric media, and the contribution $f_p(\bar{x})$ can be extended to account for these effects [Pauly et al. 2000] (see also Figure 3). This volumetric extension is also discussed in more detail in the supplementary material to Jakob et al. [2012].

The radiative transfer theory was introduced to graphics by Kajiya [1984], but is used in many areas of physics and other sciences (e.g., oceanography, geoscience, medical imaging). While fully describing the theory is well beyond our scope, we summarize here a few main concepts, and their inclusion in the path formulation.

Isotropic volumes. A homogeneous, isotropic volume (with an isotropic phase function) can be described by its *scattering coefficient* σ_s and absorption coefficient σ_a, which express the expected number of scattering/absorption events per unit ray length (and therefore have values in the range $[0, \infty)$ and units of mm^{-1}). The *extinction coefficient* σ_t is defined as $\sigma_s + \sigma_a$, and the *single-scattering albedo* as the fraction $\alpha = \sigma_s/\sigma_t$. Heterogeneous isotropic media can be described by making the coefficients spatially variant: we will write $\sigma_s(x)$ instead of σ_s, etc.

The *volume rendering equation*, which is a reformulation of the integro-differential radiative transfer equation, expresses the radiance $L(x, \omega)$ at a point x inside a volume as

$$L(x,\omega) = \int_x^y \tau(x,x') \frac{\sigma_s(x')}{4\pi} \int_{S^2} L(x',\omega') d\omega' dx' + \tau(x,y) L(y,\omega) + L_e(x,\omega). \tag{2}$$

Here $y = r(x, -\omega)$ is the first point on the boundary of the medium that is hit by a ray from x in direction $-\omega$ (so the integral is over the segment between x and y). $L(y, \omega)$ is the radiance entering the volume predicted by the standard rendering equation. Note that y could either be on a surface or on the boundary of a surfaceless volume; both cases can be handled by the path formulation. $L_e(x, \omega)$

is volume emission (always zero in our examples), and

$$\tau(y_1, y_2) = \exp\left(-\int_{y_1}^{y_2} \sigma_t(y)dy\right)$$

is the transmittance, that is, the fraction of light that passes through the line segment (y_1, y_2) without being absorbed or out-scattered.

Recursively expanding this equation yields the extension of the path formulation (similarly to the surface path formulation derivation by expanding the rendering equation [Veach 1997]). The material term $M(x_{k-1} \leftarrow x_k \leftarrow x_{k+1})$ for vertices in the volume will be equal to $\sigma_s(x_k)/4\pi$. The geometry term $G(x_k \leftrightarrow x_{k+1})$ will not contain cosine terms for vertices in the volume, since there are no cosines in (2), and it will additionally contain the factor $\tau(x_k, x_{k+1})$.

Anisotropic volumes. We support Jakob et al.'s [2010] fully anisotropic version of the radiative transfer equation, which replaces the coefficients $\sigma_s(x)$ and $\sigma_t(x)$ by direction-dependent quantities $\sigma_s(x, \omega)$ and $\sigma_t(x, \omega)$, and allows for a fully anisotropic (i.e., not rotationally invariant) phase function $f_p(\omega \leftarrow x' \leftarrow \omega')$. It can also be formulated as a pure integral equation

$$\begin{aligned} L(x, \omega) = \int_x^y \tau(x, x')\sigma_s(x', \omega) \\ \times \int_{S^2} f_p(\omega \leftarrow x' \leftarrow \omega')L(x', \omega')d\omega' dx' \\ + \tau(x, y)L(y, \omega) + L_e(x, \omega), \end{aligned} \tag{3}$$

where

$$\tau(y_1, y_2) = \exp\left(-\int_{y_1}^{y_2} \sigma_t(y, y_1 \rightarrow y_2)dy\right).$$

The path formulation can be obtained by a minor modification of the isotropic case, by setting the material term $M(x_{k-1} \leftarrow x_k \leftarrow x_{k+1}) = \sigma_s(x', \omega) f_p(\omega \leftarrow x' \leftarrow \omega')$, and using the anisotropic version of τ in the geometry term. Jakob et al.'s microflake model is one particular way to define the functions ω_s, ω_t, and f_p, which satisfies physical constraints of reciprocity and energy conservation. We use the variation from Zhao et al. [2011], which requires a fiber orientation for each spatial point and a global fiber roughness parameter; see the original papers for further information. Our current system cannot edit the phase function.

3.3 Mapping from Single-Scattering Albedos to Pixels

In full generality, every point x in the volume can be parameterized by independent scattering and extinction functions $\sigma_s(x, \omega)$ and $\sigma_t(x, \omega)$, and phase function $f_p(\omega \leftarrow x' \leftarrow \omega')$. The final appearance of the scene is a complicated, indirect function of these parameters, emerging through single and multiple scattering, shadowing, interreflection, and coupling with other surfaces in the scene. Interactive editing of all of these parameters independently does not appear tractable. To make progress on this problem, let us first consider the important subproblem of editing spatially varying single-scattering albedos $\alpha(x) \in (0, 1)$. (Note that we additionally assume the albedos themselves are not directionally dependent; therefore, we can write $\sigma_s(x, \omega) = \alpha(x)\sigma_t(x, \omega)$.) We will also mention extinction editing in Section 7.

We discretize the function $\alpha(x)$ into n piecewise homogeneous cells $C_1, \ldots, C_n$, defining the albedo vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. (In the following, we will use bold font for vectors of coefficients, to distinguish them from scalar values. We will also assume monochromatic edits, remembering that an extension to RGB is straightforward.) These cells can have any shape, and other volume parameters (extinction, phase function) do not have to be aligned with the cells in any way. The only requirement is that the cells are disjoint, and their union covers all points where volume scattering could occur.

Let us further fix a pixel, so that we can study the function $I : \mathbb{R}^n \rightarrow \mathbb{R}$ that computes the intensity of the given pixel as a function of the albedo vector. Using Eq. (1), and making the dependence on $\boldsymbol{\alpha}$ explicit, we can write

$$I(\boldsymbol{\alpha}) = \int_\Omega f(\bar{x}, \boldsymbol{\alpha})d\mu(\bar{x}).$$

Recall that $f(\bar{x}, \boldsymbol{\alpha})$ is a product of pixel weight, light emission, geometry, and material terms. Crucially, the only terms that depend on the albedo vector are the material terms for path vertices that correspond to volume scattering events. If x_k is a scattering event that occurs in cell C_j, then

$$M(x_{k-1} \leftarrow x_k \leftarrow x_{k+1}) = \alpha_j \sigma_t(x_k, x_k \rightarrow x_{k-1}) f_p(x_{k-1} \leftarrow x_k \leftarrow x_{k+1}).$$

This means that we can write the function $I(\boldsymbol{\alpha})$ as

$$I(\boldsymbol{\alpha}) = \int_\Omega a(\bar{x}) \prod_{j=1}^{n} \alpha_j^{b_j(\bar{x})} d\mu(\bar{x}), \tag{4}$$

where $a(\bar{x})$ collects terms that do not depend on the albedo vector, and $b_j(\bar{x})$ is an integer specifying the number of times the path $\bar{x}$ scatters in the cell with albedo α_j. In the following section, we will outline our technique for interactive editing of $\boldsymbol{\alpha}$ based on Eq. (4).

4. ALBEDO EDITING

In general, Eq. (4) is an integral over paths of all possible lengths; however, if we restrict the path length to say 100, the expression becomes a polynomial of degree 100 in n variables. Storing this polynomial would require data of size $\Theta(n^{100})$, which is clearly not within practical reach. Our goal is to avoid the *curse of dimensionality* in this problem. (Note that paths this long can still be important for multiple scattering. Reducing the number to 10 leads to darkening, and degree-10 polynomials would still be intractable.)

Clearly, we need to settle for approximations to compress this function; but no standard methods to achieve such compression of nonlinear, high-dimensional functions are available. This means we have to utilize some special property of $I(\boldsymbol{\alpha})$. Fortunately, such properties exist: an image containing a piece of clothing will not change completely when the albedo of a small patch is increased. Instead, the pixels local to this patch will most significantly increase intensities, while other pixels will very slightly increase due to global effects of light diffusion and inter-reflection.

This suggests similarity to relighting: if instead of albedos we were projecting a pattern and changing the values of the projector pixels, the resulting linear change in image pixel values would have a similar local-global effect. Of course, this linearity does not hold for albedo edits, but suggests that some linear (or affine) approximation may be successful.

4.1 A First-Order Approximation

Let us first consider a simple Taylor expansion of the function $I(\boldsymbol{\alpha})$. This does not yet satisfactorily solve the editing problem, but does provide an important stepping stone. The expansion is computed for a fixed assignment of albedos $\boldsymbol{\alpha_0}$, which we will term the *expansion point*. Intuitively, $\boldsymbol{\alpha_0}$ represents a "starting" set of albedo coefficients, and we would like to achieve our edits by linear

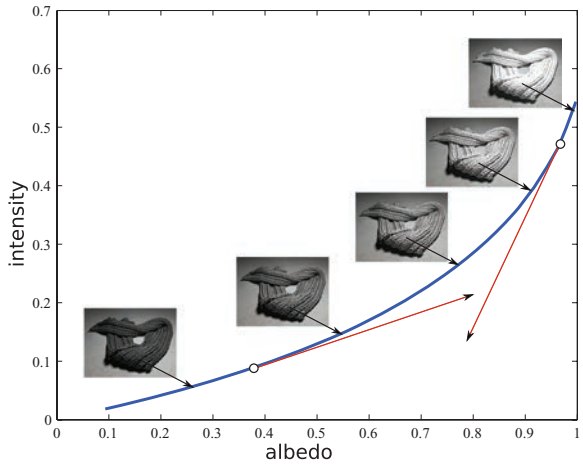

Fig. 4. The nonlinearity of the albedo-intensity curve is caused by longer paths becoming important as albedo approaches 1 (i.e., absorption approaches zero). First-order extrapolation will often result in underestimated illumination; this can be severe especially if using a high-albedo expansion point.

extrapolation from this point. A first-order approximation at $\boldsymbol{\alpha_0}$ is simply

$$I(\boldsymbol{\alpha}) \approx I(\boldsymbol{\alpha_0}) + \nabla I(\boldsymbol{\alpha_0}) \cdot (\boldsymbol{\alpha} - \boldsymbol{\alpha_0}),$$

where $\nabla I(\boldsymbol{\alpha_0})$ is the gradient of $I(\boldsymbol{\alpha})$ at $\boldsymbol{\alpha_0}$ (an n-dimensional row vector). Its entries can be computed as the first partial derivatives of Eq. (4).

$$\frac{\partial I}{\partial \alpha_i} = \frac{1}{\alpha_i} \int_\Omega a(\bar{x}) b_i(\bar{x}) \prod_{j=1}^{n} \alpha_j^{b_j(\bar{x})} d\mu(\bar{x}) \tag{5}$$

Note how the path formulation was highly useful in computing this partial derivative; this would have required much more involved manipulation if we used the original rendering equation and radiative transfer equation. Furthermore, the form of the integral is very similar to the original one, which suggests that a modification of the same Monte Carlo algorithm used for computing $I(\boldsymbol{\alpha})$ (i.e., the standard path-tracer) can be used to compute $\nabla I(\boldsymbol{\alpha})$. The gradient can even be computed if I contains effects other than the volume itself: interreflection, arbitrary surface BRDFs, arbitrary illumination, even depth-of-field effects. Note that the storage of this first-order approximation is tractable, since we just need to store a single n-dimensional vector $\nabla I(\boldsymbol{\alpha_0})$ and a single value $I(\boldsymbol{\alpha_0})$ per pixel.

4.2 Matching the Albedo-Intensity Curve

How about the accuracy of the simple affine approximation? The problem is that, even in the homogeneous case of $n = 1$ (i.e., a single editable albedo coefficient), it cannot capture the nonlinearity of the *albedo-intensity curve*: the graph of pixel intensity as a function of single-scattering albedo. This curve is a well-known feature of radiative transfer, and is caused by longer paths becoming important as albedo approaches 1 (i.e., absorption approaches zero). Figure 4 illustrates the issue: extrapolation from a low-albedo expansion point will result in underestimated illumination (though in practice this may look acceptable, except it will lose the diffusion look typical of high-albedo volumes). Even worse, starting from a high-albedo expansion point will result in an approximation that underestimates much more severely and can even become negative.

The preceding discussion of the homogeneous albedo-intensity curve suggests an improvement to the simple first-order method: could we somehow make our approximation match this curve, while still retaining the tractability of the linear approximation? In the following, we will formalize these two seemingly vague and conflicting constraints.

Let us define the *homogeneous diagonal* as the set of homogeneous albedo vectors.

$$\{\boldsymbol{\alpha} | \alpha_1 = \cdots = \alpha_n\}$$

We can now formalize the albedo-intensity curve as a function $I_h : (0, 1) \to \mathbb{R}$ that is the restriction of the full intensity I onto the homogeneous diagonal.

$$I_h(\alpha) = I(\alpha, \ldots, \alpha)$$

Using this notation, we can express our constraints on the desired approximation J as follows.

(1) We require the approximation J to be exact along the homogeneous diagonal. That is, for any homogeneous albedo vector $\boldsymbol{\alpha_h} = (\alpha_h, \ldots, \alpha_h)$, we require that

$$J(\boldsymbol{\alpha_h}) = I(\boldsymbol{\alpha_h}) = I_h(\alpha_h).$$

(2) We require that the partial derivatives of I and J match at a given expansion point $\boldsymbol{\alpha_0} = (\alpha_0, \ldots, \alpha_0)$ on the homogeneous diagonal.

$$\left.\frac{\partial J}{\partial \alpha_i}\right|_{\boldsymbol{\alpha_0}} = \left.\frac{\partial I}{\partial \alpha_i}\right|_{\boldsymbol{\alpha_0}}$$

The following theorem states that we can find an approximation J of a particularly simple and tractable form that satisfies these constraints.

THEOREM 1. *An approximation J that satisfies both of the preceding conditions can be defined as*

$$J(\boldsymbol{\alpha}) = I_h(\mathbf{w} \cdot \boldsymbol{\alpha}), \tag{6}$$

where the (row) weight vector $\mathbf{w}$ is proportional to the gradient $\nabla I(\boldsymbol{\alpha_0})$, normalized such that $\sum_{i=0}^{n} w_i = 1$.

PROOF. The first condition (equality of values) has to be true along the whole homogeneous diagonal, that is, for all

$$\boldsymbol{\alpha_h} = (\alpha_h, \ldots, \alpha_h)^T, \quad \alpha_h \in (0, 1).$$

This is easy to see: since the weight vector is normalized and all elements of $\boldsymbol{\alpha_h}$ are equal to a fixed scalar α_h, we will simply have $\mathbf{w} \cdot \boldsymbol{\alpha_h} = \alpha_h$. In other words, a blend (i.e., a linear combination with weights summing up to 1) of a set of identical values will produce the same value again.

To prove the second condition (equality of all partial derivatives), we first find the partial derivative of J at any $\boldsymbol{\alpha}$, by a simple application of the chain rule

$$\frac{\partial J}{\partial \alpha_i} = I_h'(\mathbf{w} \cdot \boldsymbol{\alpha})\, w_i,$$

where I_h' is the derivative of the one-dimensional function I_h. Furthermore, by definition of $\mathbf{w}$ (a normalized gradient), we have

$$w_i = \frac{\left.\frac{\partial I}{\partial \alpha_i}\right|_{\boldsymbol{\alpha_0}}}{\sum_{j=1}^{n} \left.\frac{\partial I}{\partial \alpha_j}\right|_{\boldsymbol{\alpha_0}}}.$$

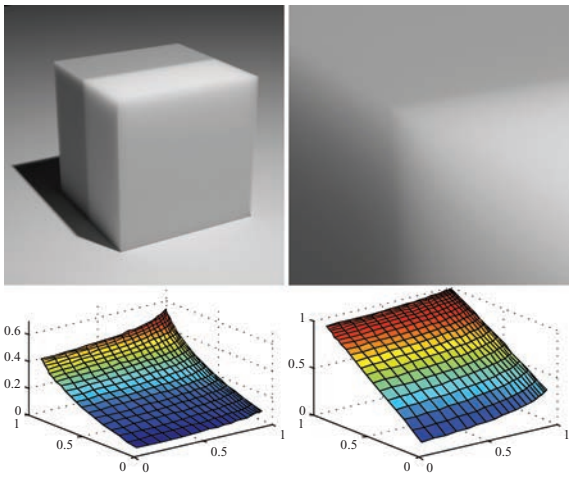

Fig. 5. Top: a scene with two volumetric cubes, and a detail of the edge between these cubes. Bottom left: A single pixel's intensity is a two-dimensional, continuous, increasing function of the albedos α_1 and α_2. A single linear approximation cannot capture the function well. Bottom right: The precomputed curve method effectively "undoes" the curvature along the homogeneous diagonal, making the function much more amenable for linear approximation.

Now we can see the equality of partial derivatives at the expansion point $\boldsymbol{\alpha_0} = (\alpha_0, \ldots, \alpha_0)^T$.

$$\frac{\partial J}{\partial \alpha_i}\Big|_{\boldsymbol{\alpha_0}} = I_h'(\alpha_0) w_i = \left(\sum_{j=1}^{n} \frac{\partial I}{\partial \alpha_j}\Big|_{\boldsymbol{\alpha_0}} \right) w_i = \frac{\partial I}{\partial \alpha_i}\Big|_{\boldsymbol{\alpha_0}}$$

This concludes the proof. □

An alternative view of Eq. (6) is that we are using a first-order expansion of the function $I_h^{-1}(I(\boldsymbol{\alpha}))$, which is much "flatter" than $I(\boldsymbol{\alpha})$. Figure 5 illustrates the insight behind the technique in two dimensions. A volumetric cube with fixed extinction is split into two halves, with albedos α_1 and α_2. The intensity of the center pixel in the detail view (top right) is a two-dimensional function $I(\alpha_1, \alpha_2)$, plotted in the bottom left. Clearly, any single first-order Taylor expansion of this function is only accurate in a small neighborhood of the expansion point. In contrast, the function $I_h^{-1}(I(\alpha_1, \alpha_2))$ (bottom right) is much flatter. In fact, it is equal to the identity function along the homogeneous diagonal $\alpha_1 = \alpha_2$, and off-diagonal values are relatively well approximated by first-order expansion points along the homogeneous diagonal.

4.3 Our Algorithm

We have found an approximation that satisfies two desirable accuracy conditions, but how do we turn it into a practical material editing system? This system will have two components, precomputation and runtime evaluation, similar to existing precomputed light transport systems. The reader can also refer to Figure 6.

Precomputation. For a given expansion point $\boldsymbol{\alpha_0}$, precomputation consists of these three steps.

(P1) Precompute the one-dimensional curve $I_h(\alpha)$ for discrete values of α. We use 10 points, distributed nonuniformly so that higher-albedo regions are more densely sampled. At each point, we use 1,024 to 4,096 Monte Carlo samples.

(P2) Compute the gradient $\nabla I(\boldsymbol{\alpha_0})$ using a modified path-tracing algorithm, at a *single* expansion point α_0. We use 512 to 1,024 samples per pixel, which leads to matrices with tractable storage and reasonably low noise. The effects of the choice of α_0 on the result are discussed in Section 6.

(P3) Compute the weight vector $\boldsymbol{w}$ by normalizing the gradient such that $\sum_{i=1}^{n} w_i = 1$.

The aggregate result of this precomputation, for all pixels, is a set of images of the homogeneous volume for k discrete albedo values, together with an $m \times n$ weight matrix $\boldsymbol{W}$, the rows of which are the normalized gradients of pixel intensity with respect to albedo coefficients. In our scarf result, we use $m = 640 \times 480 = 307,200$ pixels, $n = 262,401$ albedo coefficients, and $k = 8$ discrete points for the homogeneous curves $I_h(\alpha)$.

Storage cost. The storage requirements of the Jacobian matrix add up to an $m \times k$ matrix of homogeneous images, and an $m \times n$ weight matrix. The latter is by far the more costly component to store. The problem is that $\boldsymbol{W}$ is dense: the derivative of a given pixel's intensity with respect to any material coefficient can be nonzero, due to the global nature of light transport. How do we make the precomputation, storage, and runtime evaluation of this large matrix tractable?

Truncation of small elements to zero can cause subtly wrong results. Furthermore, adapting techniques from precomputed radiance transfer also did not lead to satisfactory results in our experiments. Harmonic bases are insufficient to control local detail. Wavelets are better, but still required hundreds of preserved coefficients in our tests, and imposing the wavelet basis on a set of albedo cells can be tricky to implement, if the cells are not on a uniform grid (such as our scarf and rose scenes). We address this challenge by noting that any Monte Carlo path generation approach naturally results in a sparse approximation to the weight matrix. Any path will scatter in a finite number of cells, often much less than 100 (our maximum path length). Using, say, 512 to 1024 paths per pixel will thus lead to an approximation in which at most a few hundred elements are nonzero. Storing such a matrix (e.g., in compressed row format) becomes tractable. One may wonder if the noise resulting from this approximation is not too high, but we have observed that in practice this is quite acceptable: the noise in I_h is more important for the final quality, and we are free to choose any number of samples there. This simple technique works quite well in our tests, and is scalable to large values of n (over 1.5 million in our rose example).

Runtime evaluation. The result of Theorem 1 can be interpreted as a two-step algorithm for each pixel.

(R1) Compute $\alpha_w \leftarrow \mathbf{w} \cdot \boldsymbol{\alpha}$.

(R2) Compute pixel value by table lookup: $J(\boldsymbol{\alpha}) \leftarrow I_h(\alpha_w)$.

Step (R1) computes a single, *local* albedo value α_w by multiplying the sparse weight matrix $\boldsymbol{W}$ by the target vector of albedos $\boldsymbol{\alpha}$. This local albedo can be thought of as a linear blend of the target albedos: a first-order approximation to the "optimal" blend that can be computed as $I_h^{-1}(I(\boldsymbol{\alpha}))$. Note that α_w is also defined for pixels that are not on the volume itself but also, for example, on the floor plane; it is used to compute global illumination effects due to the volume on these pixels. Step (R2) then performs a simple linearly interpolated table lookup to compute the value $I_h(\alpha_w)$.

5. DATA PRECOMPUTATION

The precomputation component of our system is based on an underlying path-tracing algorithm, though alternative Monte Carlo or

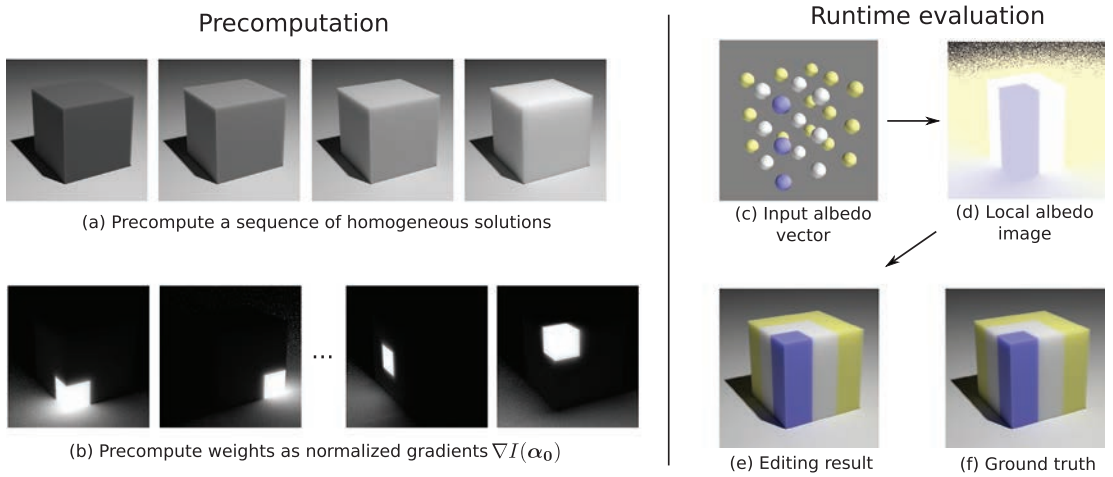

Fig. 6. An overview of our algorithm. We precompute two pieces of data: a sequence of discrete samples of the homogeneous albedo-intensity mapping $I_h(\alpha)$ (a), and the per-pixel weight vectors $\boldsymbol{w}$, which are equal to the gradients $\nabla I(\boldsymbol{\alpha_0})$ at an expansion point $\boldsymbol{\alpha_0}$, normalized so their sums are 1. (b) At runtime, a desired vector of albedo values (c) is multiplied by the weight matrix to yield a per-pixel "local" albedo approximation; (d) (note that the noise occurs mostly in areas of almost no variation of I_h and does not pose a problem). Finally, a per-pixel lookup of I_h is performed to arrive at the result (e).

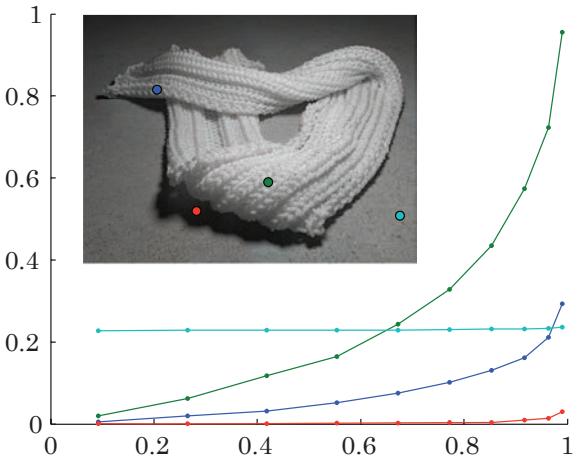

Fig. 7. The precomputed albedo-intensity curves I_h for different pixels. We use $k = 10$ discrete samples, spaced with decreasing step size towards higher albedo.

photon mapping approaches would also be possible. Recall that the precomputation phase consists of computing the $m \times k$ matrix $\boldsymbol{I}$, which consists of k images of a homogeneous material, and an $m \times n$ matrix $\boldsymbol{W}$ of weights, where m is the number of pixels and n is the number of material coefficients, and the runtime phase consists of multiplying the albedo vector by $\boldsymbol{W}$ and applying the homogeneous lookup table to the resulting local albedo image.

Matrix $\boldsymbol{I}$ of albedo-intensity curves can be precomputed using standard path-tracing. We use 10 discrete sample points, spaced nonuniformly with higher sampling for higher albedos; this handles homogeneous edits without objectionable artifacts (Figure 7).

The key implementation hurdle is to precompute the weight matrix $\boldsymbol{W}$ at an expansion point $\boldsymbol{\alpha_0}$. Recall that $\boldsymbol{W}$ is obtained from the Jacobian matrix, by scaling its rows so that their sums are equal to 1. This reduces the problem into one of computing the gradients, which were defined in Eq. (5) as

$$\frac{\partial I}{\partial \alpha_i} = \frac{1}{\alpha_i} \int_\Omega a(\bar{x}) b_i(\bar{x}) \prod_{j=1}^{n} \alpha_j^{b_j(\bar{x})} d\mu(\bar{x}).$$

Moreover, computing the value of the pixel (which the underlying path-tracing already does) was expressed in Eq. (4) as

$$I(\boldsymbol{\alpha}) = \int_\Omega a(\bar{x}) \prod_{i=1}^{n} \alpha_j^{b_j(\bar{x})} d\mu(\bar{x}).$$

Note that these two expressions have a very similar form: the gradient is given by the same path integral, except weighted by the exponent $b_i(\bar{x})$ of albedo coefficient α_j on this path. Given a path $\bar{x}$, the value $b_i(\bar{x})$ can be computed by counting how many of its vertices lie in cell C_i. We will adapt a standard path-tracing technique with explicit direct illumination connections to compute the gradients. We compute all values of the gradient at once, instead of running a separate integration for every (i, j)pair, which would be impractically slow.

At a high level, path-tracing can be described as an algorithm that generates a single subpath through the scene from the camera (using importance sampling at every vertex if possible), and connects every vertex on the subpath to a randomly chosen vertex on a light source. Figure 8 shows an example, where a single subpath $x_0x_1x_2x_3$, together with the light connections, can be treated as three separate paths $x_0x_1l_1$, $x_0x_1x_2l_2$, and $x_0x_1x_2x_3l_3$. For each of these paths $\bar{x}$, and at every vertex, we have to find the albedo cell C_j that the vertex belongs to, and increase the value of the corresponding Jacobian by 1. Furthermore, this update has to be weighted by the path contribution $f_i(\bar{x}, \hat{z}_0)$ and divided by the probability density of the path being generated.

We are not aware of a similar technique that would attempt computation of gradients with respect to material coefficients, in the presence of full light transport. Gradients have been widely used in graphics for interpolation [Ward and Heckbert 1992; Igehy 1999] but only in the spatial domain. An elegant method to compute any derivatives of a piece of code is described by Piponi [2004] but

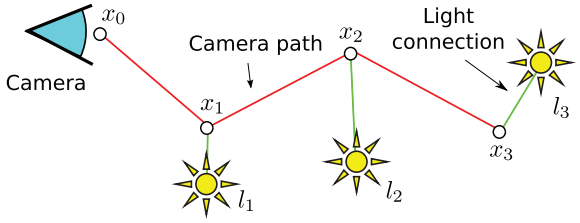

Fig. 8. Path tracing with direct illumination generates a single subpath through the scene from the camera (red), and connects every vertex on the path to a randomly chosen vertex on a light source. This can be treated as three separate paths $x_0x_1l_1$, $x_0x_1x_2l_2$, and $x_0x_1x_2x_3l_3$.

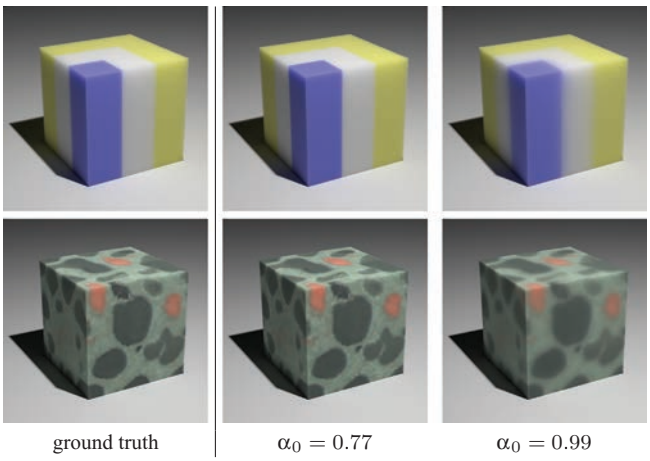

Fig. 9. The effect of different expansion point albedos on the nature of the approximation.

would not scale to n partial derivatives, where n can be over a million.

6. RESULTS

We implemented the data precomputation in the Mitsuba C++ framework [Jakob 2010], and ran it on a 32-core Intel system with hyper-threading (64 threads). The interactive material update is implemented in CUDA and runs on an Nvidia GTX 480 GPU. Figure 11 shows several examples of edits achieved with our algorithm, and Figure 10 lists the performance, precomputation times, and other information about our examples.

6.1 Illustration on Heterogeneous Cubes

An important question remains: for which $\boldsymbol{\alpha_0}$ should we precompute the first-order expansion, that is, the weight matrix $\boldsymbol{W}$? Clearly, this will impact the approximation error, but how exactly? Figure 9 shows a comparison of the effects of our approximation with a path-traced reference image, for a cube with 3^3 cells (top) and a cube with 64^3 cells (bottom). We show the effect for expansion points (i.e., matrices $\boldsymbol{W}$) computed at three different values of α_0: 0.77, 0.88, and 0.99. The mean free path of the material is 1mm and the size of the cube is 10cm; with this amount of scattering, the contribution of cells deep inside the material will be minimal, and our gradient precomputation captures this.

As we can observe, the overall brightness and colors match the reference for both choices of α_0, but the choice influences the amount of diffusion in the heterogeneous pattern. For the 3^3 cube, the edge between materials shows an increasing amount of color leaking as the expansion point albedo increases; similarly, for the 64^3 cube, the high-frequency detail in the pattern becomes blurred. At $\alpha_0 = 0.99$, there is significant blurring, which produces an interesting diffusion effect, but this diffusion is incorrect unless the albedo coefficients being blurred are all close to 0.99 (in which case the cube would look almost uniformly white anyway). Similarly, low α_0 may lead to edges that are too sharp. This suggests that if only a single matrix $\boldsymbol{W}$ is used, one should use a medium value of α_0, for example, in the range 0.7 to 0.9. We have chosen a value of $\alpha_0 = 0.772$ for the more complex results shown shortly. However, in Section 7, we will see that we can also combine multiple expansion points.

Scene	Cube 3^3	Cube 64^3	Dragon	Scarf	Rose
# pixels	512 ×512	512 ×512	640 ×480	640 ×480	640 ×480
# editable coefficients	27	262,144	262,144	262,401	1,430,565
I_h MC samples	1,024	1,024	4,096	1,024	1,024
I_h albedo samples	10	10	10	10	10
I_h data size	10 MB	10 MB	12 MB	12 MB	12 MB
precomputation	31 min	31 min	158 min	356 min	98.3 min
$\boldsymbol{W}$ MC samples	1,024	1,024	1,024	1,024	512
$\boldsymbol{W}$ data size	28 MB	566 MB	829 MB	273 MB	1197 MB
precomputation	1.3 min	1.8 min	3.6 min	29 min	12 min
runtime evaluation	6 ms	67 ms	81 ms	48 ms	272 ms

Fig. 10. Performance, precomputation times, and memory requirements of our examples. Precomputation is on a single 32-core system with hyper-threading (i.e., a total of 64 threads), while runtime is on an Nvidia GTX 480. Note the two components we precompute: the homogeneous curves I_h and the weight matrices $\boldsymbol{W}$. The data is precomputed for a single color channel; RGB variation is added at runtime by edits that differ in color channels.

6.2 More Complex Examples

We present three more complex scenes that demonstrate the practical capabilities of our editing system. The images are shown in Figure 11. We also refer the reader to the supplemental video accessible through the ACM Digital Library, which demonstrates that these edits can indeed be accomplished interactively.

Dragon. This scene demonstrates several additional features that are not present in the preceding cube examples. A stencil dragon mesh is used to cut out a part of the 64^3 voxel grid. The size of the matrix $\boldsymbol{W}$ will be identical to the aforesaid cube example: 307,200 rows and 262, 144 columns. The voxels outside of the stencil will not have any influence on the appearance: the corresponding columns of the Jacobian matrix will be zero (in the relighting analogy, this is similar to lights that are pointed away from the scene). Also note that the light diffusion through edges and creases is correctly matching the ground truth: this is not trivial to achieve using BSSRDF-based or diffusion-based methods, which are derived under assumptions that are false for thin features. This example also demonstrates the possibility of editing scattering materials in the presence of glossy (or any other) BRDFs. The edits to the material fully interact with the scene, and are clearly reflected in the floor.

Scarf. The scarf model has been created by Jonathan Kaldor [Kaldor et al. 2008]. This scene shows the capability of our method to edit heterogeneous single-scattering albedo parameters also in

Fig. 11. Example edits computed at interactive rates with our algorithm. Each image is compared with a path-traced reference, showing the high accuracy of our approximation. Note the range of effects that our technique captures: shadowing, interreflection within the volume and onto the floor (e.g., in the dragon example), transmission through thin objects (e.g., petals of the rose) and light diffusion (in all examples).

the presence of anisotropic scattering. The scarf is defined as a high-resolution grid, where every voxel is assigned a "density" (extinction) value and a local fiber orientation. To reduce storage requirements, the data is stored using a two-level hierarchical grid. These values, in conjunction with a global "fiber shininess" value and a heterogeneous albedo field editable by our system, define the phase function at every point in space using the microflake model of Jakob et al. [2010]. We defined the albedo separately using a Voronoi-diagram-like point cloud representation. We define a set of 262,401 points along the yarns (in our implementation these correspond to the vertices of the simulation mesh, but this is not a requirement), and subdivide the space R^3 into cells C_j based on the nearest point. We are then able to assign separate single-scattering albedos to each cell C_j. Note how our algorithm is able to replicate complex features of the illumination, including diffusion, self-shadowing, and inter-reflection.

Rose. This example presents the most complex stress test for our method. The light transport within the rose is a complicated combination of reflection, transmission, and scattering, and long light paths carry a large fraction of the energy. We represent the petals of the rose as solid meshes with very small but nonzero thickness: the diameter of the flower is about 10cm, while the thickness of the petals is 0.25mm. The material is represented as an isotropically scattering volume with an extinction coefficient $\sigma_t = 0.5\text{mm}^{-1}$. Figure 11 (top right) shows the rose with a homogeneous albedo setting of 99%, 95%, and 75% for red, green, and blue channels. The yellowish glow in the center emerges naturally; there is no albedo variation that would cause this. Our technique can be used to change these values interactively, which could simplify the search for the settings that yield the desired subtle effects. Moreover, we also allow for a texture to be mapped onto the flower (Figure 11, bottom right). Similar to the scarf example, a Voronoi-like point cloud scheme is used to cover the petals with points whose albedos are editable. We show results with almost 1.5 million albedo coefficients, showing the scalability of our technique.

Discussion. Our method is essentially a first-order Taylor expansion, done in a way that preserves accuracy along the homogeneous diagonal. Therefore, the best accuracy can be expected for material vectors close to the homogeneous diagonal, but in fact our results are quite good even outside of this range, for example, in the bottom scarf and rose images. The numerical error in the bottom scarf and rose images in Figure 11, measured as the L2-norm of the difference divided by the L2-norm of the reference, is 7% and 6.1% respectively. Part of this is due to Monte Carlo noise: for comparison, the difference between two different reference images is 3.9% and 2.5%, respectively. Some differences beyond noise can be perceived, but appear acceptable for an editing application.

7. EXTENSIONS

Combining expansion points. While a single expansion point chosen at a medium albedo (e.g., $\alpha_0 = 0.77$) works very well for many practical edits, we have also explored combining multiple expansion points with different values of α_0. As noted previously, when using a single expansion point, the diffusion of high-frequency texture detail will depend on the albedo at which the expansion point was computed; the problem nonlinearity causes more diffusion in areas where average albedo is high.

Figure 12 shows a cube of 64^3 voxels with a marble-like texture. We computed the expansion points at 3 different albedos α_0 (top row). For each of these images and for every pixel, an intermediate result of the editing computation is the local albedo value α_w,

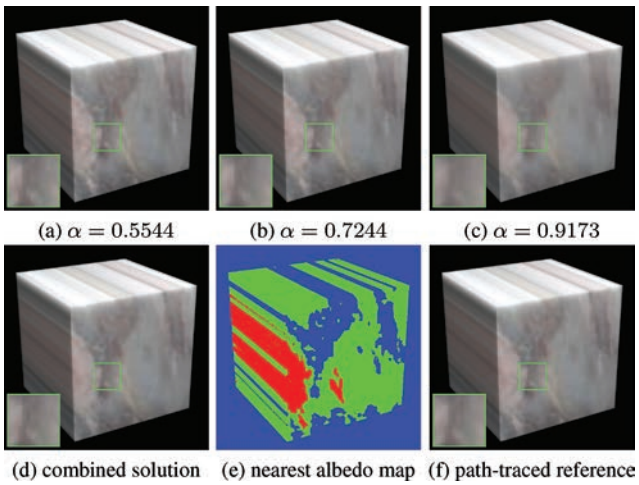

Fig. 12. Combining expansion points computed at 3 different albedos α_0. Each expansion point is most accurate for blending albedos close to its own α_0. Images computed with each of these expansion points separately are shown in (a) through (c). For each pixel, we find the difference between predicted albedo and α_0, and choose the expansion point where this difference is minimal (d). We then pick the respective pixels from (a) through (c) and combine them into (e).

as discussed in Section 4.3. This suggests a simple approach for combining the values: simply choose the α_w that is closest to the respective α_0 of the expansion point. This way we can construct an *expansion point index map*, specifying which expansion point should be used for each pixel of the image, which can be seen in Figure 12(d). The result of this combined computation (e) is visually closer to the path-traced reference solution than any of the single expansion point images (a) through (c), but not dramatically so.

Overall mean free path editing. We have also explored editing the mean free path (extinction coefficient, or optical "density") of a material. Note that a large heterogeneous change to the density could be used to change geometry, for example, creating holes in an object, or even moving it to a different location, with the associated updates in shadow and inter-reflection. This appears intractable in practice, but we have addressed the important subproblem of editing an overall multiplier of the mean free path. More precisely,

$$\sigma_t(x, \omega) = \rho\sigma_t(x, \omega)',$$

where ρ is a multiplier editable within some range $[\rho_{min}, \rho_{max}]$, and $\sigma_t(x, \omega)'$ is a base extinction (which can be both heterogeneous and anisotropic, as in the scarf model). In practice, this requires precomputing (for each pixel) a two-dimensional function $I_h(\alpha, \rho)$, instead of the one-dimensional $I_h(\alpha)$ used previously.

$$J(\boldsymbol{\alpha}, \rho) = I_h(\boldsymbol{w}(\rho) \cdot \boldsymbol{\alpha}, \rho)$$

In practice, this means two changes to the algorithm from Section 4.3. First, the function $I_h(\alpha, \rho)$ will be discretized into a two-dimensional array of images, for different values of α and ρ. Second, the weight vector $\boldsymbol{w}(\rho)$ now depends on ρ; in other words, the weight matrix can be different for different material extinctions. Precomputing many matrices along the interval $[\rho_{min}, \rho_{max}]$ would be intractable in terms of storage; we instead use a simple approximation of computing the matrices only at the endpoints ρ_{min} and ρ_{max} and interpolating the in-between weights linearly. An example

of this mean free path editing approach, applied to the scarf model, is shown in Figure 1 (right).

8. LIMITATIONS AND FUTURE WORK

Our precomputation times are still relatively high; these could be partially reduced by a GPU implementation. A speedup could be achieved by improved precomputation of the I_h curves, reusing the same paths for different albedos. We could also consider heterogeneous edits to the mean free path or phase function parameters. Our approach may be adapted to settings where analytical BSSRDF approximations are used instead of full Monte Carlo simulations (e.g., skin). The method may also be suitable as the forward component of an inverse system, such as that used for fabrication (a user would set the materials of an object designed for 3D printing, and observe an interactive preview of the fabricated appearance).

9. CONCLUSION

We presented an algorithm for editing heterogeneous single-scattering albedos in volumetric materials with multiple scattering. We achieve interactive edits with high accuracy for examples such as knitted cloth and a complex flower. Our solution combines precomputed *homogeneous* edits (a problem with strong nonlinearity but only a single dimension) with large Jacobian matrices that allow for significant heterogeneity, and are analogous to light transport matrices from well-known relighting methods. The algorithm considers the full light transport within the material, including shadowing and inter-reflection; unlike analytical methods, it has no limitation to flat surfaces. We also proposed extensions for overall mean free path editing, and combining multiple Jacobian matrices. Our editing approach could become a valuable tool in scene design, as rendering research is increasingly turning towards modeling the internal structure of materials to deliver higher realism.

ACKNOWLEDGMENTS

We thank Wenzel Jakob for his excellent Mitsuba path-tracing framework and support; Jonathan Kaldor, Manuel Vargas, and Manolis Savva for the scarf model; Jiamin Bai and Michael Tao for their help with video production; James O'Brien for compute hardware; and our reviewers, whose comments have led to substantial improvements of the article.

REFERENCES

ARVO, J. 1993. Linear operators and integral equations in global illumination. In *SIGGRAPH Course Notes 42*.

BEN-ARTIZ, A., EGAN, K., DURAND, F., AND RAMAMOORTHI, R. 2008. A precomputed polynomial representation for interactive brdf editing with global illumination. *ACM Trans. Graph. 27*, 13:1–13:13.

BEN-ARTIZ, A., OVERBECK, R., AND RAMAMOORTHI, R. 2006. Realtime brdf editing in complex lighting. *ACM Trans. Graph. 25*, 945–954.

BOUTHORS, A., NEYRET, F., MAX, N., BRUNETON, E., AND CRASSIN, C. 2008. Interactive multiple anisotropic scattering in clouds. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games (i3D'08)*. E. Haines and M. Meguire, Eds., 173–182.

DORSEY, J. O., ARVO, J., AND GREENBERG, D. P. 1995. Interactive design of complex time-dependent lighting. *IEEE Comput. Graph. Appl. 15*, 2, 26–36.

IGEHY, H. 1999. Tracing ray differentials. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'99)*. 179–186.

JAKOB,W. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

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*, 53:1–53:13.

JAKOB, W. AND MARSCHNER, S. 2012. Manifold exploration. A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. 31*, 4, 58:1–58:13.

KAJIYA, J. T. 1986. The rendering equation. *SIGGRAPH Comput. Graph. 20*, 143–150.

KAIJYA, J. T. AND VON HERZEN, B. P. 1984. Ray tracing volume densities. *SIGGRAPH Comput. Graph. 18*, 165–174.

KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2008. Simulating knitted cloth at the yarn level. *ACM Trans. Graph. 27*, 3, 1.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. All-Frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22*, 376–381.

NIMEROFF, J. S., SIMONCELLI, E., AND DORSEY, J. 1994. Efficient rendering of naturally illuminated environments. In *Proceedings of the 5th Eurographics Workshop on Rendering*. 359–373.

PAULY, M., KOLLIG, T., AND KELLER, A. 2000. Metropolis light transport for participating media. In *Proceedings of the Eurographics Workshop on Rendering Techniques*. 11–22.

PIPONI, D. 2004. Automatic differentiation, c++ templates, and photogrammetry. *J. Graph. GPU Game Tools 9*, 4, 41–55.

RAMAMOORTHI, R. 2009. *Precomputation-Based Rendering*. Now Publishers.

SCHRODER, K., KLEIN, R., AND ZIKE, A. 2011. A volumetric approach to predictive rendering of fabrics. *Comput. Graph. Forum. 30*, 4, 1277–1286.

SEQUIN, C. H. AND SMYRL, E. K. 1989. Parameterized ray-tracing. *SIGGRAPH Comput. Graph. 23*, 307–314.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 527–536.

SONG, Y., TONG, X., PELLACINI, F., AND PERS, P. 2009. Subedit: A representation for editing measured heterogeneous sub surface scattering. *ACM Trans. Graph. 28*, 31:1–31:10.

SUN, X., ZHOU, K., CHEN, Y., LIN, S., SHI, J., AND GUO, B. 2007. Interactive relighting with dynamic brdfs. *ACM Trans. Graph. 26*.

VEACH, E. 1997. Robust monte carlo methods for light transport simuation. Ph.D. Thesis, Stanford University.

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In *Proceedings of the EG Symposium on Rendering Techniques*. 195–206.

WANG, J., ZHAO, S., TONG, X., LIN, S., LIN, Z., DONG, Y., GUO, B., AND SHUM, H.-Y. 2008. Modeling and rendering of heterogeneous translucent materials using the diffusion equation. *ACM Trans. Graph. 27*, 1, 1–18.

WARD, G. J. AND HECKBERT, P. S. 1992. Irradiance gradients. In *Proceedings of the Eurographics Workshop on Rendering.*

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2011. Building volumetric appearance models of fabric using micro ct imaging. In *ACM SIGGRAPH Papers*, Article 44.

Received December 2011; revised July 2012; accepted September 2012