Modeling and Rendering of Heterogeneous Translucent Materials Using the Diffusion Equation

Jiaping Wang
Microsoft Research Asia
Shuang Zhao
Shanghai Jiaotong University
Xin Tong, Stephen Lin, Zhouchen Lin
Microsoft Research Asia
Yue Dong
Tsinghua University
Baining Guo, Heung-Yeung Shum
Microsoft Research Asia and Tsinghua University

In this paper, we propose techniques for modeling and rendering of heterogeneous translucent materials that enable acquisition from measured samples, interactive editing of material attributes, and real-time rendering. The materials are assumed to be optically dense such that multiple scattering can be approximated by a diffusion process described by the diffusion equation. For modeling heterogeneous materials, we present the *inverse diffusion algorithm* for acquiring material properties from appearance measurements. This modeling algorithm incorporates a regularizer to handle the ill-conditioning of the inverse problem, an adjoint method to dramatically reduce the computational cost, and a hierarchical GPU implementation for further speedup. To render an object with known material properties, we present the *polygrid diffusion algorithm*, which solves the diffusion equation with a boundary condition defined by the given illumination environment. This rendering technique is based on representation of an object by a polygrid, a grid with regular connectivity and an irregular shape, which facilitates solution of the diffusion equation in arbitrary volumes. Because of the regular connectivity, our rendering algorithm can be implemented on the GPU for real-time performance. We demonstrate our techniques by capturing materials from physical samples and performing real-time rendering and editing with these materials.

Categories and Subject Descriptors: I.3.0 [**Computer Graphics**]: Methodology and Techniques

General Terms: Algorithms

Additional Key Words and Phrases: appearance modeling and rendering, subsurface scattering, diffusion approximation

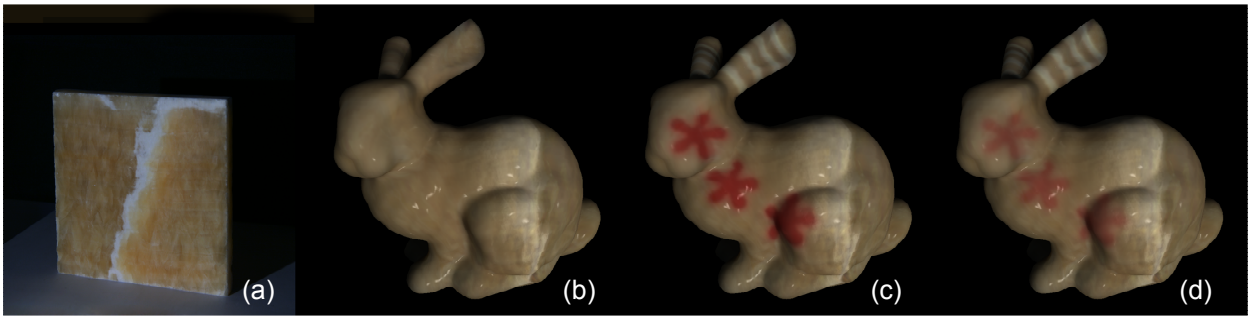

Fig. 1. Synthesis of a bunny with marble material. (a) Physical sample. (b) Bunny rendered with the acquired material model. (c) After interactive editing to add patterns. (d) After placing a pattern at a different depth.

1. INTRODUCTION

Many materials in the real world exhibit a complex appearance that arises from subsurface scattering of light. For heterogeneous translucent objects, the light transport within the material volume is determined by its geometry, the optical properties of its constituent elements, and the spatial distribution of these elements in the volume. Because of the complex effects of these various factors on subsurface scattering, models of these materials have been challenging to acquire from real objects and to render in real time. Furthermore, computational costs and/or modeling deficiencies have made interactive editing of material properties a difficult problem.

One approach for generating subsurface scattering effects is to acquire and utilize a bidirectional scattering surface reflectance distribution function (BSSRDF) [Nicodemus et al. 1977] that describes the subsurface transfer of light between two surface points. Various methods have been presented for recovering the BSSRDF of a translucent object from images captured under sampled lighting conditions and viewpoints [Goesele et al. 2004; Tong et al. 2005; Peers et al. 2006]. However, in this *surface-based* approach, physical material properties cannot be intuitively edited through modifications of 4D appearance data, which itself can be troublesome to manipulate. Moreover, rendering of surface appearance involves an integration of BSSRDF contributions from all points on the surface. Such global operations cannot be effectively implemented on the GPU without precomputation from a fixed BSSRDF.

Another approach is to construct an explicit model of scattering coefficients throughout the material volume, and then simulate subsurface scattering based on physical principles. Although a material representation can be directly modified in such a *volume-based* approach, scattering within heterogeneous translucent materials cannot be rapidly computed using previous techniques [Stam 1995; Chen et al. 2004; Li et al. 2005], thus limiting the ability to edit the material in practice. Besides the need for offline rendering, these methods do not provide a way to reliably acquire volumetric material models from real world samples.

In this paper, we propose techniques for modeling and rendering of heterogeneous translucent materials that enable acquisition from measured samples, interactive editing of material attributes, and real-time rendering. The material is represented as a discretized volume in which spatially-variant absorption and diffusion coefficients are associated with each volume element. We focus on multiple scattering and assume the material to be optically dense such that subsurface scattering becomes nearly isotropic and can be well approximated by a diffusion process [Ishimaru 1978]. This is called the diffusion approximation. In medical imaging, the diffusion approximation has been widely used to model the multiple scattering in heterogeneous human tissues [Schweiger et al. 2003; Boas et al. 2001]. For rendering participating media, Stam [1995] used the diffusion approximation to model the multiple scattering in heterogeneous clouds, where the absorption and diffusion coefficients can vary in the volume. For subsurface scattering, an analytic dipole model derived from the diffusion approximation was used by Jensen et al. [2001] for multiple scattering in homogeneous materials. In this paper, we also address subsurface scattering, but deal with the general case of multiple

scattering in heterogeneous materials with the diffusion approximation.

To model heterogeneous materials, we present the *inverse diffusion algorithm* for recovering a volumetric material model from appearance measurements by solving an inverse diffusion problem. For a given distribution of spatially-variant absorption and diffusion coefficents, the corresponding diffusion process that generates the material appearance can be expressed as a partial differential equation, defined over the volumetric elements, with the boundary condition given by the lighting environment. Acquiring a volumetric model from a material sample involves an *inverse* diffusion problem in which we search for a distribution of spatially-variant absorption and diffusion coefficents such that the corresponding diffusion process generates the material appearance that is most consistent with the measured surface appearance in captured images. Since the images record an actual material sample, a solution to the inverse diffusion problem certainly exists. This inverse problem, however, is well-known to be ill-posed, since a range of different volumetric models may have indistinguishable surface appearances [Arridge and Lionheart 1998]. Consequently, the diffusion equations and image measurements define a group of solutions. Since these solutions correspond to the same visual appearance, any solution from this group provides a valid volumetric appearance model of the given material.

Finding a solution to the inverse diffusion problem is challenging due to the nature of the inverse problem and the large number of variables involved. The inverse diffusion problem is usually solved with an iterative optimization procedure, in which each iteration requires an expensive gradient evaluation. For a volume with elements on an n^3 grid, this gradient evaluation involves $n^3 \times M$ light diffusion computations, where M is the number of image measurements. The inverse diffusion problem is also ill-conditioned numerically, which presents convergence problems for the iterative solver. To ensure stable convergence, we incorporate a regularizer on the diffusion coefficients and use an effective initialization that assigns uniform diffusion coefficents among the voxels. We additionally employ an adjoint method [Lions 1971], widely used in optimal control for gradient computation, to dramatically reduce the cost of the gradient evaluation down to $2M$ light diffusion computations. With these schemes and a GPU implementation of the diffusion computation, we show that finding a solution of the inverse diffusion problem becomes feasible for volumes of moderate size.

For rendering a volumetric model with known material properties, we present the *polygrid diffusion algorithm*, which solves a diffusion equation whose boundary condition is defined by the given illumination conditions. That multiple scattering may be modeled as a diffusion process was first observed by Stam [1995] in the context of participating media rendering. He solved the diffusion equation on a cubic volume using a regular grid and a finite difference method (FDM). Our rendering algorithm solves the diffusion equation on 3D volumes of arbitrary shape using a polygrid and an FDM. Our algorithm is centered around the polygrid representation, which facilitates the solution of the light diffusion equation in arbitrary volumes. A polygrid is a grid with regular connectivity and an irregular shape for a close geometric fit without fine sampling. The regular connectivity allows us to develop a hierarchical GPU implementation of our rendering algorithm for real-time performance. We describe how to construct a polygrid on an arbitrary 3D object, and present a technique for evaluating diffusion equations defined among the irregular intervals of polygrid nodes.

With the proposed technique, models of various materials can be readily captured and interactively transformed with adjustments of scattering properties as shown in Figure 1. This system for flexible use of real appearance data provides designers a valuable tool for creating realistic objects with the visual features of heterogeneous translucent materials. Indeed, this is the only method to date that supports real-time rendering and editing of such volumes.

2. RELATED WORK

Subsurface scattering of light within a material has typically been represented with models of radiative light transfer. Hanrahan and Krueger [1993] proposed a model based on one-dimensional linear transport theory for single scattering in layered materials. Jensen et al. [2001] presented a practical model for subsurface scattering in homogeneous materials based on an analytic dipole diffusion approximation. In [Donner and Jensen 2005], a shading model formulated from multipole theory was proposed for light diffusion in multi-layered translucent materials. We utilize the diffu-

sion equation in our work, which allows for more general modeling of multiple scattering effects in heterogeneous translucent materials.

In the following, we review related methods for acquiring subsurface scattering models, and rendering and editing of materials.

2.1 Acquisition of Subsurface Scattering Models

Subsurface scattering models of heterogeneous materials may be acquired directly from image appearance. Acquisition methods have been presented for human faces [Debevec et al. 2000], object-based models [Goesele et al. 2004], material-based models for volumes with an even distribution of heterogeneous elements [Tong et al. 2005], and material models for general heterogeneous volumes [Peers et al. 2006]. These surface-based representations are specific to the measured object or material. Although the appearance of a material could potentially be modified in these models, physical material properties cannot be edited in a meaningful way.

Models of subsurface scattering may also be acquired through estimation of scattering parameters from a material sample. Parameter estimation, however, is often confounded by multiple scattering, whose appearance arises in a complex manner from a material's scattering properties. For homogeneous materials, multiple scattering can be approximated with analytic models [Jensen et al. 2001; Narasimhan and Nayar 2003], which have greatly facilitated estimation of scattering parameters. In [Narasimhan et al. 2006], the effects of multiple scattering are avoided by diluting participating media to low concentrations, such that multiple scattering becomes negligible and scattering parameters can be solved from only single scattering. For heterogeneous, optically dense materials, multiple scattering cannot be addressed with such simplifications. Our method is the first to acquire volumetric models of heterogeneous materials.

Recently in medical imaging, special measurement devices based on time- and frequency-modulated near infrared lighting have been proposed for estimating material properties in body tissues with multiple scattering effects [Schweiger et al. 2003; Boas et al. 2001]. This approach is showing promise in its early stage of development [Gibson et al. 2005], but has limited application in computer graphics. Although its use of infrared lighting may be well-suited for tissue penetration and categorization of volume elements into particular tissue types, general scattering properties over the full visible spectrum need to be acquired for computer graphics purposes. Moreover, existing reconstruction algorithms are computationally expensive, and are practical only for coarse 3D volumes (e.g., $32 \times 32 \times 10$ in [Boas et al. 2001]) or for 2D slices. Unlike in medical imaging, computer graphics applications need not recover the actual scattering coefficients in a material volume, but instead need only to obtain a material model whose appearance is *consistent* with image measurements. This model must be acquired over the full visible spectrum and with detailed material variations for high visual fidelity. In our work, present a technique for obtaining such volumetric material models using conventional photographs. With our efficient inverse diffusion algorithm, we can successfully acquire 3D material models with a high level of detail.

2.2 Material Rendering and Editing

For rendering of BSSRDF models of subsurface scattering, several hierarchical schemes [Jensen and Buhler 2002; Carr et al. 2003] have been proposed to facilitate integration of BSSRDF contributions from all points on the surface. Since these hierarchical data structures need to be precomputed before rendering, they cannot be used in material editing. [Mertens et al. 2003] proposed a hierarchical scheme that supports real-time updates, but since this method is based on the dipole diffusion model, it can be used only for rendering homogeneous materials. In [Lensch et al. 2003], local subsurface scattering is computed by a local image filter, while the global scattering is determined from vertex-vertex transport. Although this method can provide interactive rendering speed, precomputation of the local filter and global transport makes this approach unsuitable for material editing.

Subsurface scattering has been simulated for volumetric material models using Monte Carlo methods [Dorsey et al. 1999; Pharr and Hanrahan 2000] and photon tracing [Jensen and Christensen 1998], but at a considerable expense in computation. Real-time rendering can be achieved through precomputation of light transport [Hao and Varshney 2004;

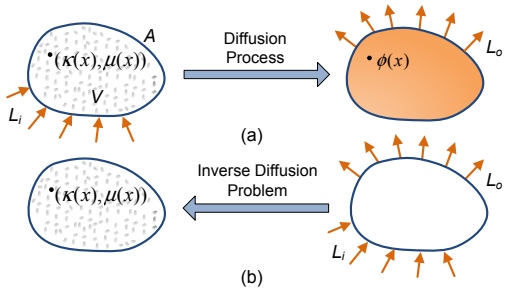

Fig. 2. Overview. (a) Rendering with forward diffusion, from the volumetric material properties ($\kappa(x)$ and $\mu(x)$) and illumination setting (L_i) to the outgoing radiance (L_o). (b) Model acquisition with inverse diffusion, from a set of illumination conditions (L_i) and measured outgoing radiances (L_o) to the volumetric material properties ($\kappa(x)$ and $\mu(x)$).

Wang et al. 2005]. However, light transport quantities that have been precomputed with respect to a given volumetric model are no longer valid after material editing.

For radiance transfer modeled as a diffusion process, the set of PDE equations may be numerically solved to determine material appearance. Multi-grid schemes and simplified volume representations have been employed to facilitate rendering of participating media in [Stam 1995]. Haber et al. [2005] used embedded boundary discretization to solve for light diffusion in object volumes of arbitrary shape, though not in real time. A finite element method (FEM) could also be used in computing light diffusion within an arbitrary object. However, FEMs require decomposition of the object into tetrahedra, whose irregular connectivity makes GPU implementation difficult. In our proposed approach, real-time evaluation of diffusion is achieved with a polygrid representation of the object volume and an adaptive hierarchical scheme for diffusion computation that has an efficient implementation on the GPU.

3. OVERVIEW

Figure 2 presents an overview of our approach. We denote the object interior as volume V and the object surface as A. The outgoing radiance $L(x_o, \omega_o)$ at a surface point x_o in direction ω_o may be computed by integrating the incoming radiance $L(x_i, \omega_i)$ from all incident directions ω_i and points x_i on surface A:

$$L_o(x_o, \omega_o) = \int_A \int_\Omega S(x_i, \omega_i, x_o, \omega_o) L_i(x_i, \omega_i)(n \cdot \omega_i) \mathrm{d}\omega_i \mathrm{d}A(x_i),$$

where n is the surface normal at x_i and $S(x_i, \omega_i, x_o, \omega_o)$ is the BSSRDF. The outgoing radiance can be divided into single- and multiple-scattering components:

$$L_o(x_o, \omega_o) = L_s(x_o, \omega_o) + L_m(x_o, \omega_o).$$

The single-scattering component $L_s(x_o, \omega_o)$ accounts for light that interacts exactly once with the medium before exiting the volume, and may be evaluated by integrating the incident radiance along the refracted outgoing ray, as described in Eq. (6) of [Jensen et al. 2001].

In our method, we focus on multiple scattering $L_m(x_o, \omega_o)$ that consists of light that interacts multiple times within the object volume, which is the dominant scattering component for optically dense materials. For highly scattering, non-emissive materials, the multiple scattering can be approximated by a diffusion process described by the following equation [Ishimaru 1978]:

$$\nabla \cdot (\kappa(x) \nabla \phi(x)) - \mu(x) \phi(x) = 0, \qquad x \in V, \tag{1}$$

with boundary condition defined on the object surface A [Schweiger et al. 1995]:

$$\phi(x) + 2C\kappa(x)\frac{\partial\phi(x)}{\partial n} = q(x), \qquad x \in A, \tag{2}$$

where $\phi(x) = \int_{4\pi} L_o(x,\omega)\mathrm{d}\omega$ is the radiant fluence (also known as the scalar irradiance or radiant flux), $\kappa(x) = 1/[3(\mu(x) + \sigma_s'(x)]$ is the diffusion coefficient, $\mu(x)$ is the absorption coefficient, and $\sigma_s'(x) = \sigma_s(1-g)$ is the reduced scattering coefficient with g being the mean cosine of the scattering angle. We define $C = (1+F_{dr})/(1-F_{dr})$, where F_{dr} is the diffuse Fresnel reflectance that is determined by the refraction index η of the material [Jensen et al. 2001]. The diffused incoming light at a surface point x is given by $q(x) = \int_\Omega L_i(x,\omega_i)(n\cdot\omega_i)F_t(\eta(x),\omega_i)\mathrm{d}\omega_i$, where $F_t(\eta(x),\omega_i)$ is the incoming Fresnel transmission term. Note that this diffusion process well describes light transport between voxels of different optical properties since the κ and μ used in Eq. (1) and Eq. (2) may vary throughout the volume.

With the diffusion approximation, the multiple scattering component of the outgoing radiance is approximated by

$$L_m(x_o,\omega_o) = -\frac{C}{\pi}F_t(\eta(x_o),\omega_o)\kappa(x_o)\frac{\partial\phi(x_o)}{\partial n}, \tag{3}$$

where $\phi(x_o)$ is computed from Eq. (1) and Eq. (2) [Schweiger et al. 1995; Jensen et al. 2001].

Our work centers on modeling and rendering multiple scattering in a heterogeneous material using the diffusion approximation. For rendering an object with known $\mu(x)$ and $\kappa(x)$ throughout the object volume V, we solve the diffusion problem with a given illumination condition $q(x)$ on the object surface A. Once the solution $\phi(x)$ is found, the multiple scattering component of the outgoing radiance can be easily evaluated using Eq. (3). We note that the diffusion equation assumes scattering to be frequent enough to be considered nearly isotropic and independent of the phase function.

In acquiring the material properties from measured appearance, we need to compute the absorption coefficients $\mu(x)$ and diffusion coefficients $\kappa(x)$ based on measured outgoing radiances $\{L_{o,m}(x,\omega_o) \mid x \in A,\ m = 0,1,\ldots M\}$ from the object surface due to multiple scattering under M different illumination conditions $\{L_{i,m}(x,\omega_i) \mid x \in A,\ m = 0,1,\ldots M\}$ on the object surface. For this purpose, we solve the inverse diffusion problem to find $\kappa(x)$ and $\mu(x)$ such that the corresponding diffusion problem, which is expressed by Eq. (1), Eq. (2), and Eq. (3), produces the outgoing radiance $L^R_{o,m}(x,\omega_o)$ that is most consistent to the measured outgoing radiance $L_{o,m}(x,\omega_o)$ under the same illumination conditions $L_{i,m}(x,\omega_i)$. The inverse diffusion problem is thus formulated as finding the values of $\kappa(x)$ and $\mu(x)$ throughout the volume that minimize the objective function

$$\sum_{m=1}^{M}\int_A\int_\Omega \left(L_{o,m}(x,\omega_o) - L^R_{o,m}(x,\omega_o)\right)^2 \mathrm{d}A(x)\mathrm{d}\omega_o. \tag{4}$$

To obtain multiple scattering components from image measurements, a cross-polarization approach as described in [Debevec et al. 2000] may be employed. We instead utilize an image acquisition scheme described in Section 4.1 that minimizes the presence of single scattering and surface reflections in the image data.

4. INVERSE DIFFUSION ALGORITHM FOR ACQUISITION OF MATERIAL MODEL

To acquire the volumetric material model of a real object, we obtain images of the object under different illumination conditions and then solve the inverse problem of light diffusion on the multiple scattering components. In solving the inverse diffusion problem, we search for the volumetric model $(\mu(x),\kappa(x))$ whose forward diffusion solution is most consistent with the acquired images. This procedure is described in the following subsections.

4.1 Data Capture

We use a Canon 30D digital camera with a 17-45mm lens to record images of a material sample that is illuminated by an Optoma DLP projector with a 4500:1 contrast ratio. In our experiments, the material samples are all block-shaped

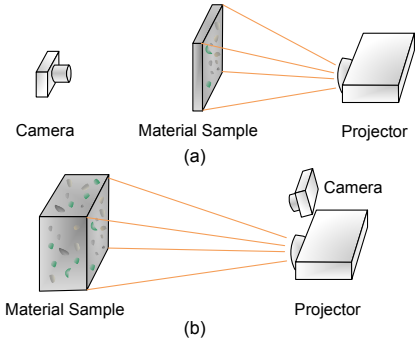

Fig. 3. Two experimental setups used for data acquisition. (a) back-lighting setup for thin material samples. (b) front-lighting for thick material samples.

and represented as a regular grid with $n \times m \times l$ sample points ($n \geq m \geq l$) on the grid nodes. As shown in Figure 3, we utilize two setups depending on the thickness of the sample. In both setups, we position the sample so that one of the $n \times m$ faces is perpendicular to the optical axis of the projector. For thin material samples ($n, m \gg l$), the camera is placed facing the sample from the opposite side, such that the sample is imaged with back-lighting. For thick material samples with little transmission of light through the volume, we position the camera beside the projector as done in [Peers et al. 2006]. We will refer to the side of the sample facing the camera as the front face.

The camera and projector are calibrated prior to image acquisition. For radiometric calibration of the camera, we apply the method of [Debevec and Malik 1997]. Geometric calibration of both the camera and projector is done with the technique in [Zhang 1999], where for the projector we project a chessboard pattern onto different planes. The white balance of the camera is calibrated with respect to the projector, based on the projection of a white image onto a Macbeth Color Checker^{TM} chart with known albedos. The color chart is also used in measuring the black level of the projector. To avoid interference effects from the projector's color wheel, we utilize exposure times of at least 1/30 second.

In illuminating the sample, we subdivide the face that receives direct lighting into 4×4 regions, and separately project light onto each region while capturing an image sequence of the complete sample with a fixed aperture and variable exposure times ranging from 1/30 to 8 seconds. Using the method in [Debevec and Malik 1997], we construct an HDR image from the image sequence for each illumination condition. Vignetting effects from the projector are minimized by illuminating the sample using only the center of the projector images. This capture process typically takes about half an hour.

With this capture process, we obtain images of multiple scattering data. In the thin-sample setup, the back-lighting is assumed to scatter multiple times before exiting from the front face, such that captured images contain only multiple scattering. In the setup for thick samples, we utilize image data only from surface points that are not directly illuminated by the projector. The appearance of these points is considered to result only from multiple scattering. Since the projector and camera are aligned perpendicularly to the material sample in both configurations, we can disregard the Fresnel transmittance effects on the measured multiple scattering, and drop the dependence on ω_o in Eq. (4). For all examples in this paper, we follow [Jensen et al. 2001; Tong et al. 2005] and assume that the refraction index of all materials η is 1.3. For Eq. (2) and Eq. (3), this value of η is used to derive the constant $C = 2.1489$.

4.2 Volumetric Model Acquisition

For each captured image and corresponding lighting condition, we map onto each grid node on the front and back faces its incoming light intensity and measured outgoing radiance. The material model is then acquired by solving for the scattering parameters (κ and μ) of each node in the volume that would result in image appearances most consistent

Set initial material properties: $\vec{\kappa}_0$, $\vec{\mu}_0$
Set initial search direction: $\vec{d}_0 = -\vec{z}(\kappa_0, \mu_0)$ and $\vec{p}_0 = \vec{d}_0$
Repeat following steps until $f_M < \varepsilon$
Compute gradient $\vec{z}(\kappa_t, \mu_t) = \left(\frac{\mathrm{d} f_M(\vec{\kappa}, \vec{\mu})}{\mathrm{d}\kappa(x)}, \frac{\mathrm{d} f_M(\vec{\kappa}, \vec{\mu})}{\mathrm{d}\mu(x)} \right)$
Set $p_t = -\vec{z}(\kappa_t, \mu_t)$
Update search direction $\vec{d}_t = \vec{p}_t + \beta \cdot \vec{d}_{t-1}, \quad \beta = \max\left(\frac{\vec{p}_t^T (\vec{p}_t - \vec{p}_{t-1})}{\vec{p}_{t-1}^T \vec{p}_{t-1}}, 0 \right)$
Golden section search λ' by $\min_{\lambda'} \left[f_M \left((\vec{\kappa}_t, \vec{\mu}_t) + \lambda' \vec{d}_t \right) \right]$
Update solution $(\vec{\kappa}_{t+1}, \vec{\mu}_{t+1}) = (\vec{\kappa}_t, \vec{\mu}_t) + \lambda' \vec{d}_t$

Table I. Conjugate gradient based algorithm for minimizing f_M.

with the measured data. With the M measured images of the material sample, we thus aim to minimize the following objective function:

$$f_M(\vec{\kappa}, \vec{\mu}) = \sum_{m=1}^{M} f_m(\vec{\kappa}, \vec{\mu}) + \lambda \sum_{x \in V} \|\nabla \kappa(x)\|^2,$$

where $f_m(\vec{\kappa}, \vec{\mu}) = \sum_{x \in A} \left(L_{o,m}(x) - L^R_{o,m}(x) \right)^2$ measures the consistency between the measured outgoing radiance $L_{o,m}(x)$ from all front face points x and the outgoing radiance $L^R_{o,m}(x)$ that is computed from the estimated scattering parameters with the illumination condition of image m. Note that in f_m we drop the dependence on ω_o that is present in Eq. (4) because of our imaging configuration. The vectors $\vec{\kappa}$ and $\vec{\mu}$ represent the set of diffusion and absorption coefficients defined over all the grid nodes. Since model acquisition is ill-conditioned with respect to κ, we add a regularization term $\sum_{x \in V} \|\nabla \kappa(x)\|^2$ to the objective function, weighted by a constant λ set to $1e-5$ in our implementation.

To minimize f_M, we employ the conjugate gradient algorithm outlined in Table I. From an initialization of $\vec{\kappa}$ and $\vec{\mu}$, we first compute the gradient of f_M with respect to $(\vec{\kappa}, \vec{\mu})$ over the set of measured images. The search direction is then updated with the Polak-Ribiere method [Press et al. 1992]. Subsequently, we perform a golden section search to find the optimal step length λ' along the search direction. Finally, we update $\vec{\kappa}$ and $\vec{\mu}$ using the computed gradient $\vec{z}(\kappa, \mu)$ and λ'. These steps are iterated to update $\vec{\kappa}$ and $\vec{\mu}$ until the objective function falls below a threshold set to $\varepsilon = 10^{-4} \times \sum_{x \in A} [L_{o,m}(x)]^2$ in our implementation. This optimization is performed separately on the RGB channels.

To initialize the scattering parameters in this optimization, we solve for the volumetric material model under the assumption that it is homogeneous, i.e., all the grid nodes have the same μ, κ. Since there exist only two unknowns in this case, they can be quickly computed using the conjugate gradient procedure with user-specified initial values.

A key step in conjugate gradient optimization is the computation of the f_M gradient relative to the unknown κ and μ values at each grid node. Since the diffusion equation has no analytic solution, we compute the gradients numerically. A straightforward approach for gradient computation is to perturb each of the variables and obtain the resultant change in objective function value. One forward diffusion simulation would then be necessary to compute each gradient. Although this method is feasible for a system with few parameters (e.g., a homogeneous volume), it is impractical for arbitrary heterogeneous volumes which have a large number of unknowns. Specifically, model acquisition for an $n \times m \times l$ grid with M measurements would require $2 \times n \times m \times l \times M$ forward diffusion simulations for each iteration, clearly a prohibitive expense.

4.3 Adjoint Method For Gradient Computation

To significantly expedite gradient computation, we take advantage of the adjoint method [Giles and Pierce 1999], a technique that has been widely used in optimal control [Lions 1971]. We describe here how to directly use the adjoint method in our application, and give derivation details in Appendix A.

To use the adjoint method in our solution, we first define the adjoint equation of the original diffusion equation as

$$\nabla \cdot (\kappa(x)\nabla\varphi(x)) - \mu(x)\varphi(x) = 0, \qquad x \in V, \tag{5}$$

with boundary condition defined on the surface A:

$$\varphi(x) + 2C\kappa(x)\frac{\partial\varphi(x)}{\partial n} = \frac{2C}{\pi}(L_{o,m}(x) - L^R_{o,m}(x)), \qquad x \in A, \tag{6}$$

where $(L_{o,m}(x) - L^R_{o,m}(x))$ is the difference between the measured outgoing radiance $L_{o,m}(x)$ from all frontal sample points x and the outgoing radiance $L^R_{o,m}(x)$ that is computed from the diffusion equation with the illumination condition q_m of image m. Given φ, the gradient of f_M with respect to κ and μ at each grid point is computed by

$$\begin{aligned} \mathrm{d}f_M(\vec{\kappa},\vec{\mu})/\mathrm{d}\kappa(x) &= \sum_{m=1}^{M} \nabla\varphi_m(x)\cdot\nabla\phi_m(x) - 2\lambda\Delta\kappa(x), \\ \mathrm{d}f_M(\vec{\kappa},\vec{\mu})/\mathrm{d}\mu(x) &= \sum_{m=1}^{M} \varphi_m(x)\phi_m(x), \end{aligned} \tag{7}$$

where $\phi_m(x)$ is determined from the diffusion equation with the illumination condition q_m of image m.

In contrast to the original diffusion equation, the adjoint method utilizes "virtual" illumination to define the boundary condition. This virtual illumination $(2C/\pi)(L_{o,m}(x) - L^R_{o,m}(x))$, which may be negative, and ϕ are computed from the diffusion equation using the actual illumination condition. With the virtual illumination, we solve the adjoint equation for φ, and then determine the gradient of f_M relative to κ and μ using Eq. (7). Using the adjoint method, only $2M$ forward diffusion simulations are needed for gradient computation. Because of its computational efficiency, the adjoint method has also been used in [McNamara et al. 2004] for gradient computation in fluid control.

4.4 GPU-based Diffusion Computation

In model acquisition, forward diffusion simulations are used not only in gradient computation, but also for evaluating the objective function in the golden section search. To solve the diffusion equation on a 3D regular grid, we discretize the diffusion equation as a set of linear equations over the grid nodes using the FDM scheme in [Stam 1995]:

$$\begin{aligned} &\sum_{j=1}^{6} \frac{1}{h^2}\kappa(v_j)\phi(v_j) - \frac{6}{h^2}\kappa(v_i)\phi(v_i) - u(v_i)\phi(v_i) = 0, \\ &\phi(v'_i) + 2C\kappa(v'_i)\frac{\phi(v'_i) - \phi(v'_j)}{h} = q(v'_i), \end{aligned} \tag{8}$$

where v_j denotes one of six nodes directly connected to interior node v_i. h represents the distance between two neighboring nodes. $q(v'_i)$ denotes the incoming radiance at boundary node v'_i. And v'_j is the closest interior node to v'_i along the inward normal direction.

This linear system can be numerically solved using the relaxation scheme described in [Stam 1995], which involves considerable computation and is the bottleneck in model acquisition. For efficient processing, we present a GPU-based multiresolution scheme that simulates forward diffusion in the pixel shader on grid values of κ, μ, and q packed into separate 2D textures. This GPU-based method can be regarded as a regular-grid version of the rendering algorithm in Section 5, where we provide further details.

In solving the diffusion equation on the GPU, we upload all the relevant data from main memory to texture memory, and then output the radiant fluence results from the frame buffer back to main memory. The remaining optimization computations are all executed on the CPU. Despite some overhead for data transfer, an appreciable overall reduction in computation costs is obtained through GPU acceleration.

5. POLYGRID DIFFUSION ALGORITHM FOR RENDERING AND EDITING

After acquiring a material model from a real sample, a volume of arbitrary shape can be formed with this material using the mapping techniques described in [Chen et al. 2004; Porumbescu et al. 2005]. These approaches map the material properties into a shell layer at the object surface, and construct the inner core volume by synthesizing a user-specified material texture or interpolating from the inner boundary of the shell layer using mean value coordinates [Ju et al. 2005].

With a given lighting condition and the material properties defined throughout the object volume, the subsurface scattering effects from the object can be rendered in a three-pass process. In the first pass, we compute the incoming radiance on the object surface, based on shadow map visibility for directional or point lighting, or from precomputed radiance transfer techniques [Sloan et al. 2002; Ng et al. 2003] for environment lighting. In the second pass, we render the multiple scattering effects by simulating light diffusion inside the object volume with the incident radiance on the surface as the boundary condition. Then, a single scattering term and surface specular reflections from the incoming illumination is computed. The final rendering result is obtained by adding all of these components together.

To efficiently solve for light diffusion on the GPU, we extend the FDM scheme on regular volumetric grids to handle a polygrid defined in the object volume. A polygrid is a grid with regular 6-connections among evenly distributed nodes inside the volume, and with boundary nodes that are aligned to the object surface and are each connected to one interior node along the inward normal direction. With the polygrid representation of the object volume, we discretize the light diffusion equation and its boundary condition into a system of linear equations:

$$\begin{aligned} &\sum_{j=1}^{6} w_{ji}\kappa(v_j)\phi(v_j) - \left(\sum_{j=1}^{6} w_{ji}\right)\kappa(v_i)\phi(v_i) - u(v_i)\phi(v_i) = 0, \\ &\phi(v_i') + 2C\kappa(v_i')\frac{\phi(v_i') - \phi(v_j')}{d_{ji}} = q(v_i'), \end{aligned} \tag{9}$$

where v_j denotes one of six nodes directly connected to interior node v_i with a weight w_{ji} for the Laplacian operator, as described in Appendix B. d_{ji} represents the distance between a boundary node v_i' and the closest interior node v_j' along the inward normal direction, and $q(v_i')$ denotes the incoming radiance at surface node v_i'.

In the remainder of this section, we describe how to construct the polygrid of an object volume and how to solve the diffusion equation on the GPU. We also present a hierarchical scheme for accelerating GPU evaluation of light diffusion on a polygrid.

5.1 Polygrid Construction

The steps in constructing a polygrid model of the object volume are shown in Figure 4. We first manually assemble a polycube [Tarini et al. 2004] of similar topology that approximates the volume. Within the cubes, we form regular grids of equal resolution, and connect the grids between adjacent cubes. The interior nodes directly linked to boundary nodes on the edges or corners of the cube have connectivity to multiple boundary nodes, which may lead to artifacts in the light diffusion computation. We address this problem by removing certain nodes and adjusting the links to obtain single connections to boundary nodes. As shown in Figure 4 (g) and (h), we examine axis-aligned 2D slices of the grid and utilize different grid adjustment schemes depending on the grid convexity in each slice. This procedure yields a polygrid defined in the polycube.

We then map this polygrid to the object volume. To find a mapping, we first determine a projection of the polycube surface onto the object surface, using a PolyCube-Map [Tarini et al. 2004] or other mesh cross-parameterization method (e.g., [Kraevoy and Sheffer 2004; Schreiner et al. 2004]). The boundary nodes of the polygrid are then mapped to the object surface and adjusted to obtain an even distribution [Turk 1992]. After that, the interior nodes directly connected to the boundary nodes are placed within the object volume at a distance d along the inward normal directions, where d is one-tenth the average distance between connected boundary nodes on the object surface. The

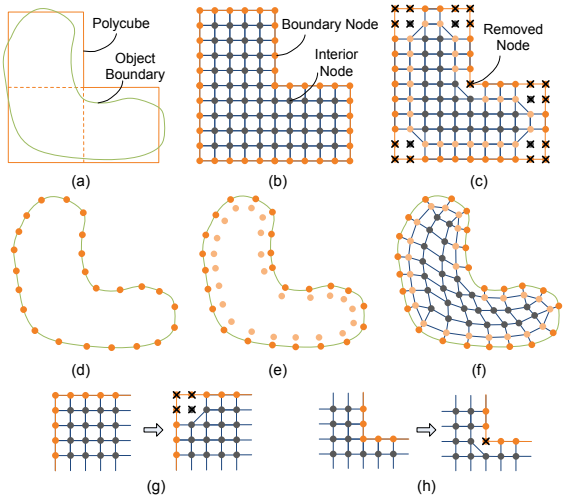

Fig. 4. A 2D illustration of polygrid construction. (a) Creating a polycube that approximates the object. (b) Generating a grid in the polycube. (c) Modifying the corner nodes. (d) Projecting boundary nodes to the object surface. (e) Mapping nodes connected to boundary nodes to the object volume. (f) Computing the final polygrid in the object volume. (g) and (h) are two schemes used to modify irregular corner nodes on 2D slices of the grid.

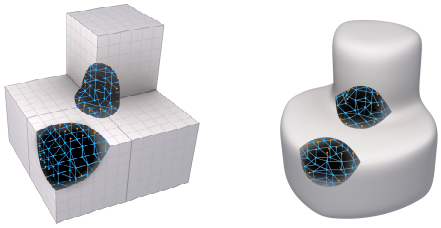

Fig. 5. The polygrid constructed in a 3D model.

close placement of these nodes to the boundary nodes is intended for accurate handling of the boundary condition. The remaining interior nodes are then positioned within the volume in a manner that minimizes the variance of distances between connected nodes: $\min \sum_{i \in interior} Var(\{\|v_i - v_j\| : j \bowtie i\})$, where $Var(\cdot)$ denotes the variance of a set of scalars, v_i is the 3D position of node i, and $j \bowtie i$ indicates that node j is connected to node i. Figure 4 illustrates this construction procedure in 2D. In principle, a conformal mapping [Gu and Yau 2003] should be used to preserve the orthogonality of the original grid connections and minimize distortion (see Appendix B). However, this remains a challenging problem for 3D volumes, so in practice we utilize the presented variance minimization scheme which we have found to yield acceptable solutions.

This construction scheme maintains the regular connectivity of nodes and produces locally uniform distributions of interior grid nodes in the object volume. As exemplified in Figure 5, all the interior grid nodes of the polygrid are 6-connected, and each boundary node is connected to exactly one interior node. The connectivity between the boundary grid nodes is not used in rendering and can be ignored in the diffusion computation.

5.2 GPU-based Polygrid Diffusion Computation

With the constructed polygrid, we build a system of linear equations for light diffusion. The material properties for each grid node are sampled from the object volume, and the incoming illumination is computed for boundary nodes.

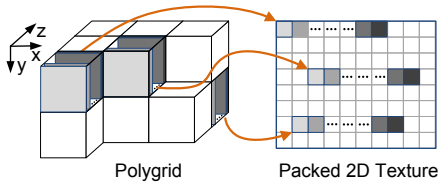

Fig. 6. Flattening a polygrid into a packed 2D texture.

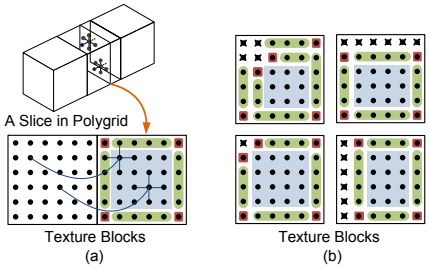

Fig. 7. Geometric primitives for nodes with different texture access patterns. (a) Three types of nodes, rendered using different geometric primitives. Each region is rendered by one primitive. (b) Geometric primitives for some texture blocks with removed nodes.

Although a general purpose GPU-based linear system solver could be used for computation [Krüger and Westermann 2003; Bolz et al. 2003], we have designed a more efficient GPU implementation that is specific to diffusion computation on a polygrid.

In this method, the polygrid material parameters are packed into a set of 2D textures for computation on the GPU. For efficient rendering, the textures must be packed such that the connected neighbors of each node are easily accessible. Towards this end, we organize each texture according to the positions of the polygrid nodes within the original polycube. We traverse the cubes in the polycube in scanline order, and flatten the grid of each cube as shown in Figure 6. The grid in each cube is divided into 2D x-y slices, which are each treated as a texture block and ordered in the texture by increasing z value. In packing the texture, we retain the empty positions of grid nodes that were previously removed, so that the cubes have slices of equal size. Two 2D textures T_κ and T_μ are created for the corresponding scattering parameters, and for the iterative computation we maintain two swap radiance buffers I_A and I_B that are organized in the same manner as T_κ and T_μ. In addition, we precompute the six weights for the Laplacian operator, then similarly pack this data into two textures T_{w1} and T_{w2}. The incoming radiance is also packed into a 2D texture T_l according to an access order that will be described later.

After texture packing, we solve the diffusion equations on the polygrid using the relaxation scheme in [Stam 1995]. Starting from the initial radiant fluence values ϕ_0, we iteratively update the radiant fluence values in the two radiance buffers until convergence. With the radiant fluence at each node corresponding to one pixel in the radiance buffer, this computation can be executed in the pixel shader with parameters accessed from the textures. To reduce texture fetches in the pixel shader, we store $\phi' = \kappa\phi$ in the radiance buffer. In each step, the radiant fluence values are updated as

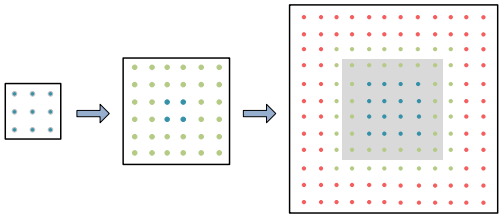

Fig. 8. A 2D illustration of the adaptive hierarchical technique used in diffusion computation. In this scheme, we fix the values of the blue nodes in the middle level after initialization from the coarsest level. The blue and green nodes at the finest resolution are also fixed after initialization. Nodes in the gray region are not used in the diffusion computation and are removed from textures.

follows:

$$\phi'_{n+1}(v_i) = \frac{\sum_{1\le j\le 6} w_{ji}(v_i)\phi'_n(v_j)}{\mu(v_i)/\kappa(v_i) + \sum_{1\le j\le 6} w_{ji}(v_i)},$$

$$\phi'_{n+1}(v'_i) = \frac{q(v'_i)\kappa(v'_i)\kappa(v'_j)d + 2C\kappa^2(v'_i)\phi'_n(v'_j)}{\kappa(v'_j)d + 2C\kappa(v'_i)\kappa(v'_j)},$$

where right-hand-side operators of the form $f(\cdot)$ involve a texture access, and the radiance buffer for ϕ'_n is used as the texture while the other radiance buffer is used as the rendering target for ϕ'_{n+1}.

As shown in Figure 7, there exist three types of nodes/pixels in the radiance buffer, each with different texture access patterns for reaching connected nodes. We render each type of node using a different geometric primitive, represented by colored regions in the figure. For a (blue) node that lies in the interior of a texture block, four of its connected neighbors in the polygrid are also adjacent neighbors in the 2D texture, while the other two neighbors can be found with the same offset value in other texture blocks. We update the values of these nodes by rendering a quadrilateral with the texture offsets of the two non-adjacent neighbors as vertex attributes. After rasterization, this offset information can be interpolated from the vertices to each pixel in the quad. In a similar manner, the (green) nodes on each texture block edge are rendered with a line, where three neighbors are adjacent in the texture, and the texture offsets of the other three are stored as line vertex attributes. The (red) nodes of each texture block corner are rendered with points, with the texture offsets of all six neighbors stored as vertex attributes. Slices that contain removed nodes can also be rendered using these three primitives. All of these geometric primitives and their vertex attributes can be precomputed and loaded into graphics memory before rendering. Since the surface boundary nodes and the interior nodes are processed differently, we render their corresponding geometric primitives in two separate passes with different pixel shaders. After completing this computation, we calculate the output radiance on the surface by updating the boundary nodes in the radiance buffer as $L(v'_i) = F_t(x_o,\omega_o)[\phi'(v_i) - q(v_i)\kappa(v_i)]/[2\pi\kappa(v_i)]$. These boundary node values are then used as a texture for surface vertices in the final pass.

With this packing and rendering scheme, the radiant fluence values are updated with ten texture fetches for interior nodes and five texture fetches for surface nodes. Compared to [Krüger and Westermann 2003; Bolz et al. 2003], our scheme avoids extra texture storage for node connectivity information and dependent texture accesses in rendering. We acknowledge that alternative schemes for packing and rendering are possible and may be better, but we nevertheless have obtained good performance with this method.

5.3 Hierarchical Acceleration

For greater efficiency in computing light diffusion, we employ a hierarchical scheme to accelerate rendering with the polygrid. In this scheme, we first construct a multiresolution polygrid in the object volume. Starting from the original

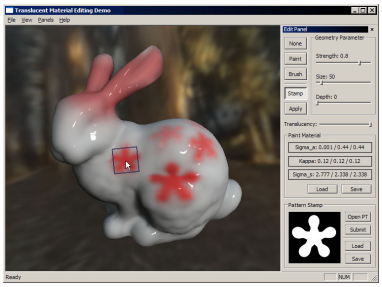

Fig. 9. User interface for material editing.

polygrid, the positions and material properties of nodes at successively coarser levels are determined by averaging the positions and material properties of its eight children at the next finer level. For nodes whose children contain removed nodes, we normalize the result by the number of existing children. Before rendering, we pack the material properties at each resolution and generate texture pyramids for T_κ, T_μ, T_{w1} and T_{w2}. Pyramids need not be generated for the radiance buffers I_A and I_B, which can simply be reused for computation at each level. During rendering, we first solve the diffusion equations at the coarsest grid level, and then use the computed radiant fluence at each node as initializations for its children nodes at the next finer level. This process iterates until a solution at the original polygrid resolution is obtained.

The hierarchical algorithm can be accelerated by employing an adaptive scheme in which light diffusion is computed to different resolutions at different depths in the volume. Since material variations deeper inside the object volume have more subtle effects on surface appearance, it is sufficient to approximate light diffusion at deeper nodes with coarser-resolution solutions. As shown in Figure 8, after we obtain the solution at a coarse resolution and copy it to a finer resolution, the radiant fluence values at nodes below a certain depth are fixed, while the nodes closer to the boundary are updated. In our implementation of this adaptive scheme, the computed resolution at different depth levels is given by the user. Texture blocks whose nodes are not used in computation at a given level are removed to save on texture storage.

In our current implementation, we do not use the V-cycle multi-grid algorithm to speed up light diffusion computation, since the multigrid algorithm cannot easily be incorporated into our adaptive scheme. Also, V-cycle multi-grid algorithms require extra texture storage for residual and temporary radiant fluence values in each level. If all grid nodes are used in light diffusion, our hierarchical solution algorithm can be regarded as a simplified N-cycle multigrid scheme without the V-cycles for each resolution [Press et al. 1992].

A favorable property of the light diffusion algorithm is that the coherence between frames can be exploited to facilitate rendering. For applications in which the lighting or material changes gradually, the rendering result of the last frame provides an excellent initialization for the current frame. With good initial values, the number of iteration steps can be significantly reduced.

5.4 Editing

With this real-time rendering system, the acquired volumetric material model can be interactively edited with real-time feedback on the modified appearance. To illustrate this capability, we developed a simple editing system shown in Figure 9. In addition to painting new values for $\mu(x)$ and $\kappa(x)$, various ways to modify existing $\mu(x)$ and $\kappa(x)$ are supported. The user can directly adjust $\mu(x)$ and $\kappa(x)$ by multiplying them with or adding them to user-supplied constants. Alternatively, the user can modulate the μ and κ values within a pattern mask using a texture. With our volumetric representation, users can also modify a material at specific depth levels. For a demonstration of a material editing session, please view the supplemental video.

In our system, all editing operations are executed as pixel operations on the GPU. We maintain extra buffers T'_κ and

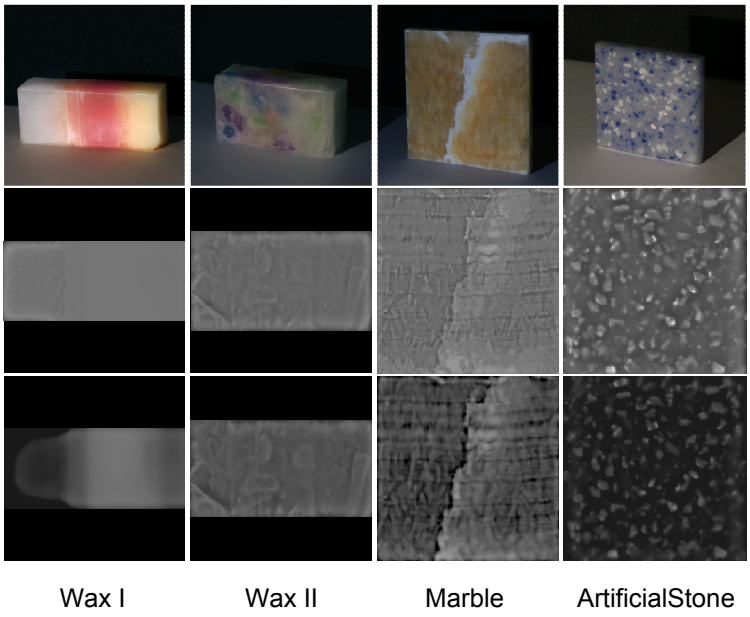

Fig. 10. Acquired material samples. Top row: real image of sample. Middle row: reconstructed κ along the surface. Bottom row: reconstructed μ along the surface. The values of κ and μ are scaled for better viewing.

T'_μ of the κ and μ textures as rendering targets for editing. In each frame, T'_κ and T'_μ are modified by user-specified operations, and then swapped to T_κ and T_μ for rendering. To support editing operations on local regions, we store the positions of grid nodes in a texture T_p. Then when the user selects a region on the screen for editing, we compute the screen projection of each grid node based on its position in the editing shader, and execute the editing operations only for the nodes within the user-specified local region. In material editing, we do not use the adaptive scheme, but instead take advantage of the coherence between frames to reduce rendering computation, which allows for more complex material editing operations to be executed on the GPU.

6. EXPERIMENTAL RESULTS

We have implemented our material acquisition and rendering system on a PC configured with an Intel Core2Duo 2.13GHZ CPU, 4GB memory, and a Geforce 8800GTX graphics card with 768MB graphics memory. The GPU-based light diffusion and rendering algorithm was implemented in the OpenGL shading language. For the GPU-based light diffusion computation used in model acquisition, we represent all parameters and computation results as 32-bit floating-point values for high precision. For light diffusion computations on the polygrid, each channel of κ and μ is quantized to 8-bits and stored together in 24-bit textures. We use 16-bit floating-point values in rendering computations, which provides sufficient precision in appearance.

6.1 Model Acquisition

Figure 10 displays the material samples used in our experiments and the acquired κ and μ values along the surfaces of the material volumes. For these samples, the grid resolutions, lighting configurations, and computation times are listed in Table II. We set the grid resolutions according to sample size and the material variability in the volume, where higher resolutions are needed to preserve the rich material variations in detailed volumes. With GPU acceleration, the reconstruction algorithm gains a fifty-fold increase in speed over a CPU-based implementation on the same platform. Because of the significant surface reflection of the marble sample, we obtain an approximate measure of its reflectance

Material	Grid Resolution	Illumination	Computation Time (hrs)
Wax I	$130 \times 53 \times 37$	back	2.0
Wax II	$140 \times 75 \times 48$	front	4.0
Marble	$256 \times 256 \times 20$	back	11.0
Artificial Stone	$128 \times 128 \times 41$	back	3.0

Table II. Acquisition settings for the different materials.

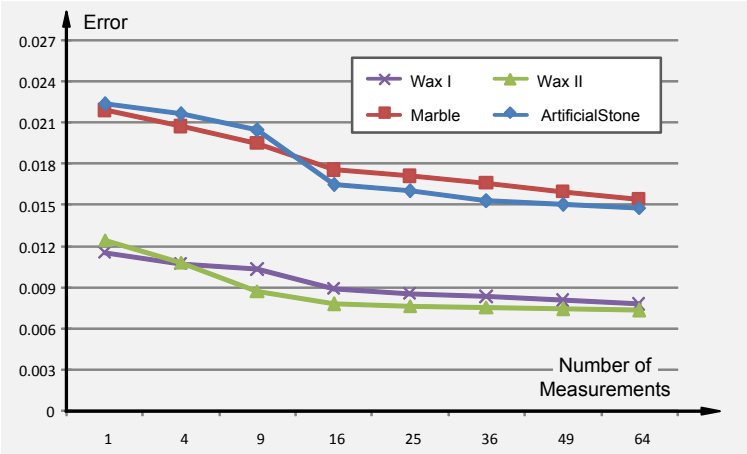

Fig. 11. Model quality vs. number of measurements. Errors of BSSRDFs computed from material volumes acquired from real material samples using different numbers of measurements.

component with the front-lighting setup and the method in [Tong et al. 2005], which records one image of the sample under uniform lighting. The diffuse surface reflectance term is obtained by subtracting the multiple scattering contribution from it.

In theory, the diffuse BSSRDF should be densely sampled to ensure that the acquired material volume generates accurate surface appearances for arbitrary illumination conditions. However, because of the redundancy in BSSRDF data, we have found that models acquired from sparsely sampled images provide good results in practice. Note that each image here corresponds to a 2D slice of the 4D BSSRDF.

To examine the relationship between the number of measurements and model quality, we applied our inverse diffusion algorithm on all material samples shown in Figure 10. We subdivide the face that receives direct lighting into $n \times n$ regions that are each separately illuminated. For different n, we acquired $n \times n$ images under different illumination as input to our algorithm. Normalized errors were then computed as $E = \sum_{x_i,x_j \in A} [R'_d(x_i,x_j) - R_d(x_i,x_j)]^2 \Big/ \sum_{x_i,x_j \in A} [R_d(x_i,x_j)]^2$, where R_d is the diffuse BSSRDF captured from the original volume, and R'_d is that computed from the acquired material volume. Figure 11 displays the errors of the material models acquired from four material samples with different numbers of measurements, $n = 1,2 \ldots 8$, which indicates that for 16 or more images, the error is comparable to that reported for the factorized BSSRDF representation in [Peers et al. 2006]. In our current implementation, we use 16 images under different illumination settings for model acquisition, which provides faithful rendering results exemplified in Figure 12.

Figure 13 illustrates the role of regularization in optimization. The regularization term adds a smoothness constraint to κ in solving the non-linear optimization of the acquisition step, while the coefficient λ is used to adjust the effects of the regularization term in non-linear optimization. Note that without a proper weight, the resulting material

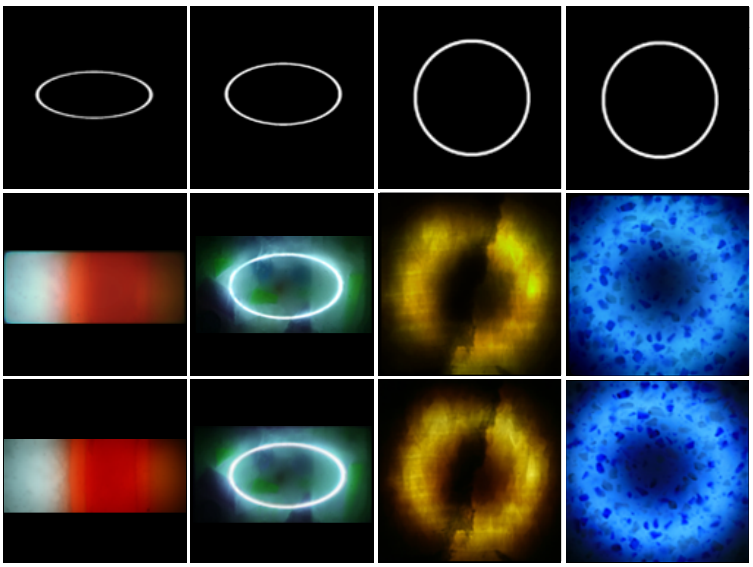

Fig. 12. Comparison of acquired models to ground truth. Top row: Illumination settings. Front lighting is applied for WaxII while back lighting is used for the other material samples. Middle row: Rendering results of volumes acquired from 16 images. Bottom row: Real images of the materials.

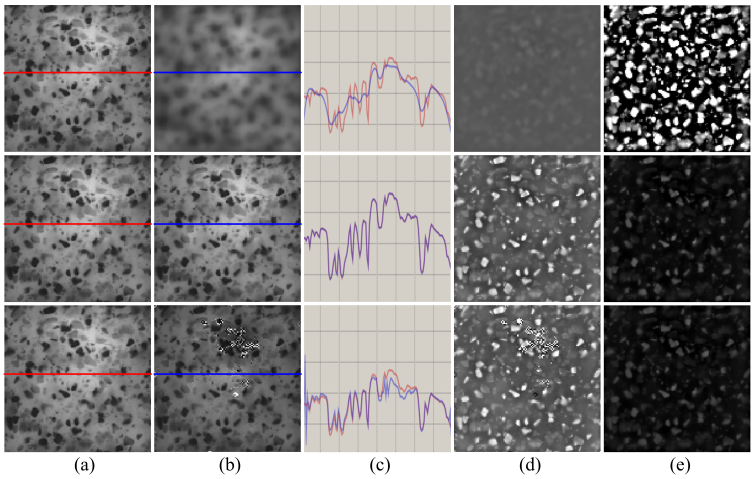

Fig. 13. Effect of the regularization parameter on optimization. Top row: result acquired with an over-weighted regularization term ($\lambda = 1.0e-4$). Middle Row: result acquired with proper regularization term ($\lambda = 1.0e-5$). Bottom Row: result acquired with an under-weighted regularization term ($\lambda = 1.0e-6$). For each row, (a) is one of the captured images used for optimization. (b) is the image computed from the acquired model with the illumination condition of image (a). In (c), we compare the values along the 1d scanline in (a) and (b), where the red line represents ground truth and the blue line represents computed values. (d) and (e) show acquired κ and μ along the top surface. The values of κ and μ are scaled for better viewing.

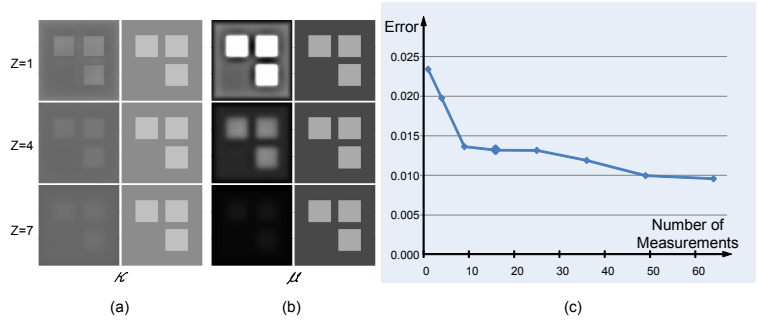

Fig. 14. Accuracy in material model acquisition. (a) Values of κ for the acquired material model (left) at different depth layers below the front face, and the ground truth (right). (b) Values of μ for the acquired material model (left) at different depths, and the ground truth (right). (c) Errors of BSSRDFs computed from acquired material model using different numbers of measurements. The material model shown in (a) and (b) is acquired from 16 measurements.

Model	Polygrid Resolution	Texture Size (MB)	Rendering Time $t_1/t_2/t_3$ (ms)	Final Speed (fps)
Bunny	$253 \times 16 \times 16 \times 16$	24.9	3.9/23.2/11.5	25.9 (34.7)
Bust	$17 \times 32 \times 32 \times 32$	18.9	2.5/19.9/7.0	34.0 (52.7)
Bird	$108 \times 24 \times 24 \times 24$	39.6	4.0/48.3/13.0	15.3 (24.0)
Hole	$36 \times 12 \times 12 \times 12$	4.8	1.5/8.9/2.1	80.0 (160.4)
Snail	$81 \times 12 \times 12 \times 12$	18.9	1.6/14.0/3.1	53.4 (106.7)

Table III. Rendering configuration and performance. For the rendering times, t_i indicates the time for the ith rendering pass. The final speed is measured both without frame coherence and with frame coherence (in parentheses).

model yields an inaccurate fit to the input data. Typically in practice, λ is experimentally set by the user. In our implementation, we found that $1.0e-5$ works well for all samples shown in the paper.

Although our approach effectively acquires a detailed appearance model of subsurface scattering, it cannot determine the actual material properties and their variations in the volume, especially deep within the volume. This results from object appearance being relatively insensitive to the material composition far beneath the surface. Figures 14(a) and (b) illustrates the properties of a material model acquired from 16 measurements of a synthetic volume. In Figure 14(c), we compare the the BSSRDFs generated from the acquired material model to the BSSRDFs rendered from the synthetic volume. The acquired material model generates a BSSRDF that can faithfully serve as an appearance model. On the other hand, it also lacks the precision needed in some applications such as solid texture mapping, where deep volumes may become exposed on the surface.

6.2 Rendering & Editing

Table III lists the polygrid resolution at the finest level, the texture size, and rendering performance for all the examples shown in the paper. The polygrid resolution is the product of the number of cubes in the polycube and the grid resolution in each cube. The polygrid resolution is determined by the resolution of the material volume such that each material voxel corresponds to one grid cell in the object volume. Our experiments show that the ploygrid model works well for materials with different scattering properties.

The texture size includes all textures used for light diffusion computation. In this table, the rendering times are broken down into the three passes, for incident light computation (t_1), light diffusion (t_2), and final rendering (t_3). For the overall rendering speed, the first number reports the frame rate without frame coherence (i.e., radiances initialized to zero), while the second number in parentheses gives the speed with frame coherence. In rendering, we use the

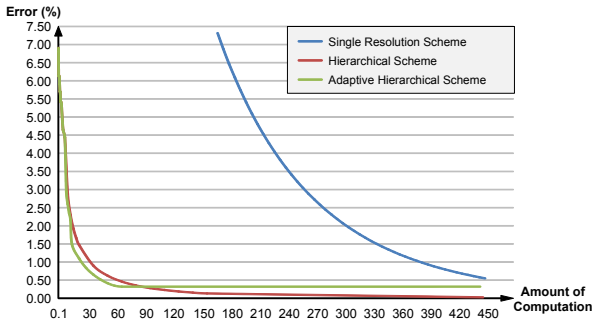

Fig. 15. Convergence speed of the three schemes. The X axis represents convergence speed in terms of 100K's of nodes that are processed.

ambient occlusion to determine visibility for environment lighting, and the shadow buffer for visibility of local lighting. A three-level polygrid with the adaptive scheme is used in the measurements for this table. In the final rendering step, specular reflections on the surface are computed with a user-specified Ward model, and single scattering is ignored. For the model with marble material, the measured surface reflectance term is also added to the multiple scattering in this step.

Figure 15 charts the convergence speed of the different rendering methods on the hole model. Here, the error of a result L is computed as $\sum_{x\in A}(L(x)-L_0(x))^2/\sum_{x\in A}L_0(x)^2$, where L_0 is the converged result precomputed without hierarchical and adaptive acceleration. A multi-resolution polygrid with three levels is used in the hierarchical schemes. The polygrid for the hole model contains 36 cubes, and the grid resolution in each cube at the finest level is $12\times12\times12$. For the hierarchical method without the adaptive scheme, we use all grid nodes in the light diffusion computation. In the adaptive scheme, we manually specify the depth of nodes that are involved in computing each resolution such that the final error is less than 0.5%. It can be seen that the hierarchical scheme can substantially improve light diffusion performance on the polygrid. With the adaptive scheme, the computation is further accelerated (a two-fold speedup in this case) to generate a result with an error below a user-specified threshold.

Figure 16 compares images of a bunny model rendered with the polygrid diffusion algorithm and the result rendered by photon mapping. The volumetric material applied to the bunny shown in Figure 16(a) (d) is modeled by the user, while the volumetric material shown in Figure 16(e) and (f) is edited from captured marble volume. All results are rendered under directional lighting, and only multiple scattering is considered. As shown in the figure, visually faithful results can be generated with our method. The small discrepancy on the bunny ear is caused by the distortion of the polygrid in that area.

Figures 17 (a) and (b) show rendering results for a bust model whose volumetric material properties are acquired from a marble sample, and a hole model generated with acquired wax material. Complex surface appearances from subsurface scattering and volumetric material variations are well preserved with our volumetric appearance model.

In Figure 18, a bird model is rendered with a reconstructed wax material and an artificial stone material. Results are shown with different viewing and lighting directions. The heterogeneity beneath the surface is well handled in our modeling and rendering technique.

Rendering results of a bunny with different translucent materials edited by the user are shown in Figure 19. Both the local scattering properties of the material and their distributions are modified in these cases. With the volumetric material model, editing of physical attributes can be done in an intuitive manner. Additional editing results are exhibited in Figure 1.

Finally, we display the rendering result of a snail in Figure 20. The volumetric material properties of the snail body are designed by an artist using our editing tool. The artist also painted the opaque snail shell.

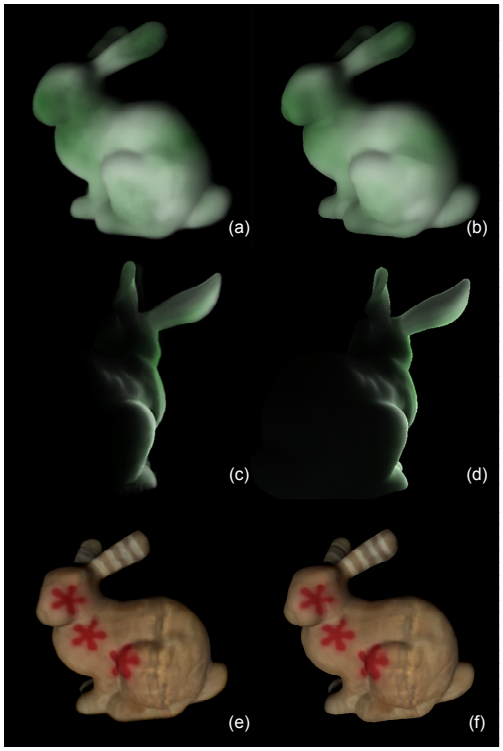

Fig. 16. Comparison of the polygrid based diffusion algorithm to photon mapping. (a)(c)(f) Results rendered by the polygrid-based diffusion algorithm. (b)(d)(g) Results rendered by photon mapping.

7. CONCLUSION

In this paper, we proposed efficient techniques based on the diffusion equation for modeling and rendering heterogeneous translucent materials. A practical scheme is presented for acquiring volumetric appearance models from real material samples, and for rendering the appearance effects of multiple scattering in real time. With this method, a user can easily edit translucent materials and their volumetric variations with real-time feedback.

In rendering, the FDM scheme used in this paper can be regarded as an approximate solution for arbitrary-shaped object volumes. With this approximation, we obtain realistic rendering results and real-time performance. For objects with fine geometric detail or complex topology, however, the presented polygrid construction scheme may produce large distortions in some regions, which can lead to rendering artifacts. We intend to address this issue in future work by examining better methods for mesh and volume parameterization.

Another problem we plan to investigate is acquisition of the single scattering effects and phase functions of heterogeneous translucent materials. In addition, we would like to extend our rendering algorithm to real-time deformation of translucent objects.

Acknowledgements

The snail in Figure 20 was modeled by Feiyang Tong. The authors thank Ming Jiang, Tie Zhou, Ruo Li and Guanquan Zhang for discussions on inverse diffusion, and the anonymous reviewers for their helpful suggestions and comments. This work was done when Shuang Zhao, and Yue Dong were visiting students in Microsoft Research Asia.

REFERENCES

ARRIDGE, S. AND LIONHEART, B. 1998. Non-uniqueness in diffusion-based optical tomography. *Optical Letters 23*, 882–884.

BOAS, D., BROOKS, D., DIMARZIO, C., KILMER, M., GAUETTE, R., AND ZHANG, Q. 2001. Imaging the body with diffuse optical tomography. *IEEE Signal Proc. Magazine 18,* 6, 57–75.

BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRÖDER, P. 2003. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. *ACM Trans. Graph. 22,* 3, 917–924.

CARR, N. A., HALL, J. D., AND HART, J. C. 2003. GPU algorithms for radiosity and subsurface scattering. In *Proc. Graphics Hardware*. 51–59.

CHEN, Y., TONG, X., WANG, J., LIN, S., GUO, B., AND SHUM, H.-Y. 2004. Shell texture functions. *ACM Trans. Graph. 23,* 3 (Aug.), 343–353.

DEBEVEC, P., HAWKINS, T., TCHOU, C., DUIKER, H.-P., SAROKIN, W., AND SAGAR, M. 2000. Acquiring the reflectance field of a human face. In *Proc. ACM SIGGRAPH*. 145–156.

DEBEVEC, P. E. AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *Proc. ACM SIGGRAPH*. 369–378.

DONNER, C. AND JENSEN, H. W. 2005. Light diffusion in multi-layered translucent materials. *ACM Trans. Graph. 24,* 3 (July), 1032–1039.

DORSEY, J., EDELMAN, A., LEGAKIS, J., JENSEN, H. W., AND PEDERSEN, H. K. 1999. Modeling and rendering of weathered stone. In *Proc. ACM SIGGRAPH*. 225–234.

GIBSON, A., HEBDEN, J., AND ARRIDGE, S. 2005. Recent advances in diffuse optical imaging. *Physics in Medicine and Biology 50*, R1–R43.

GILES, M. B. AND PIERCE, N. A. 1999. An introduction to the adjoint approach to design. In *ERCOFTAC Workshop on Adjoint Methods*.

GOESELE, M., LENSCH, H. P. A., LANG, J., FUCHS, C., AND SEIDEL, H.-P. 2004. DISCO: acquisition of translucent objects. *ACM Trans. Graph. 23,* 3, 835–844.

GU, X. AND YAU, S.-T. 2003. Global conformal surface parameterization. In *Proc. Symp. Geometry Processing*. 127–137.

HABER, T., MERTENS, T., BEKAERT, P., AND VAN REETH, F. 2005. A computational approach to simulate light diffusion in arbitrarily shaped objects. In *Proc. Graphics Interface*. 79–85.

HANRAHAN, P. AND KRUEGER, W. 1993. Reflection from layered surfaces due to subsurface scattering. In *Proc. ACM SIGGRAPH*. 165–174.

HAO, X. AND VARSHNEY, A. 2004. Real-time rendering of translucent meshes. In *ACM Trans. Graph.* Vol. 23. 120–142.

ISHIMARU, A. 1978. *Wave Propagation and Scattering in Random Media*. Academic Press.

JENSEN, H. W. AND BUHLER, J. 2002. A rapid hierarchical rendering technique for translucent materials. *ACM Trans. Graph. 21,* 3, 576–581.

JENSEN, H. W. AND CHRISTENSEN, P. 1998. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proc. ACM SIGGRAPH*. 311–320.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Proc. ACM SIGGRAPH*. 511–518.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24,* 3, 561–566.

KRAEVOY, V. AND SHEFFER, A. 2004. Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph. 23,* 3, 861–869.

KRÜGER, J. AND WESTERMANN, R. 2003. Linear algebra operators for GPU implementation of numerical algorithms. *ACM Trans. Graph. 22,* 3, 908–916.

LENSCH, H. P. A., GOESELE, M., BEKAERT, P., MAGNOR, J. K. M. A., LANG, J., AND SEIDEL, H.-P. 2003. Interactive rendering of translucent objects. In *Computer Graphics Forum*. Vol. 22(2). 195–205.

LI, H., PELLACINI, F., AND TORRANCE, K. E. 2005. A hybrid monte carlo method for accurate and efficient subsurface scattering. In *Rendering Techniques*. 283–290.

LIONS, J.-L. 1971. *Optimal control systems governed by partial differential equations*. Springer-Verlarg.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Trans. Graph. 23,* 3, 449–456.

MERTENS, T., KAUTZ, J., BEKAERT, P., SEIDEL, H.-P., AND REETH, F. V. 2003. Interactive rendering of translucent deformable objects. In *Eurographics Workshop on Rendering*, P. Dutré, F. Suykens, P. H. Christensen, and D. Cohen-Or, Eds. 130–140.

NARASIMHAN, S. G., GUPTA, M., DONNER, C., RAMAMOORTHI, R., NAYAR, S. K., AND JENSEN, H. W. 2006. Acquiring scattering properties of participating media by dilution. *ACM Trans. Graph. 25,* 3, 1003–1012.

NARASIMHAN, S. G. AND NAYAR, S. K. 2003. Shedding light on the weather. In *Proc. IEEE CVPR*. 665–672.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22,* 3, 376–381.

NICODEMUS, F. E., RICHMOND, J. C., HSIA, J. J., GINSBERG, I. W., AND LIMPERIS, T. 1977. *Geometrical Considerations and Nomenclature for Reflectance*. National Bureau of Standards (US).

PEERS, P., VOM BERGE, K., MATUSIK, W., RAMAMOORTHI, R., LAWRENCE, J., RUSINKIEWICZ, S., AND DUTRÉ, P. 2006. A compact factored representation of heterogeneous subsurface scattering. *ACM Trans. Graph. 25,* 3, 746–753.

PHARR, M. AND HANRAHAN, P. M. 2000. Monte Carlo evaluation of non-linear scattering equations for subsurface reflection. In *Proc. ACM SIGGRAPH*. 275–286.

PORUMBESCU, S. D., BUDGE, B., FENG, L., AND JOY, K. I. 2005. Shell maps. *ACM Trans. Graph. 24,* 3, 626–633.

PRESS, W. H. ET AL. 1992. Numerical recipes in C (second edition). *Cambridge University Press*.

SCHREINER, J., ASIRVATHAM, A., PRAUN, E., AND HOPPE, H. 2004. Inter-surface mapping. *ACM Trans. Graph. 23,* 3, 870–877.

SCHWEIGER, M., ARRIDGE, S., M., H., AND D.T., D. 1995. The finite element method for the propagation of light in scattering media: Boundary and source conditions. *Medical Physics 22,* 11, 1779–1792.

SCHWEIGER, M., GIBSON, A., AND ARRIDGE, S. 2003. Computational aspects of diffuse optical tomography. *Computing in Science and Engineering 5,* 6, 33–41.

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,* 3, 527–536.

STAM, J. 1995. Multiple scattering as a diffusion process. In *Euro. Rendering Workshop*. 41–50.

TARINI, M., HORMANN, K., CIGNONI, P., AND MONTANI, C. 2004. PolyCube-maps. *ACM Trans. Graph. 23,* 3, 853–860.

TONG, X., WANG, J., LIN, S., GUO, B., AND SHUM, H.-Y. 2005. Modeling and rendering of quasi-homogeneous materials. *ACM Trans. Graph. 24,* 3, 1054–1061.

TURK, G. 1992. Re-tiling polygonal surfaces. *Proc. ACM SIGGRAPH 26,* 2, 55–64.

WANG, R., TRAN, J., AND LUEBKE, D. 2005. All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Trans. Graph. 24,* 3, 1202–1207.

ZHANG, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc. IEEE ICCV*. 666–673.

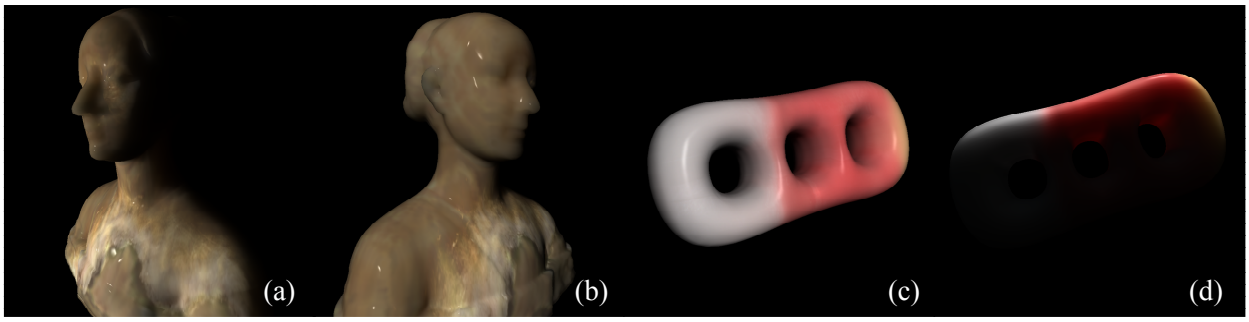

Fig. 17. Rendering results of the bust and the hole models with materials acquired from real samples. (a)(b) the bust model rendered with marble material. (c)(d) the hole model rendered with wax.

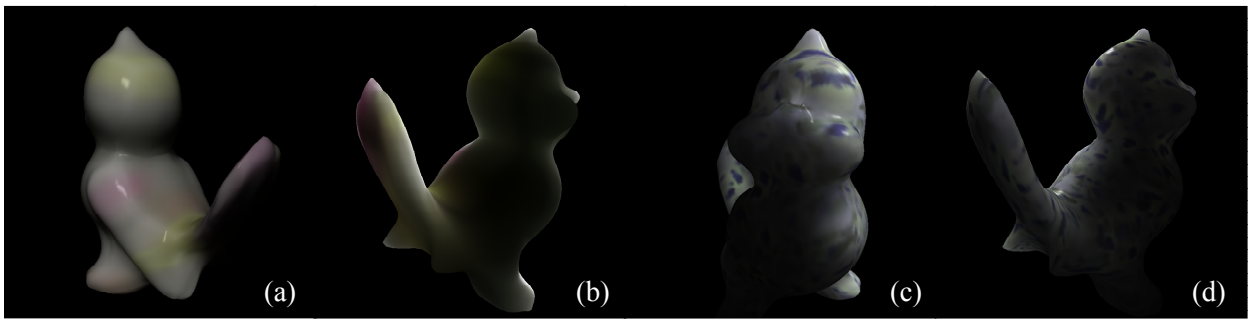

Fig. 18. Rendering results of a bird model with different materials. (a)(b) with wax material. (c)(d) with artificial stone material.

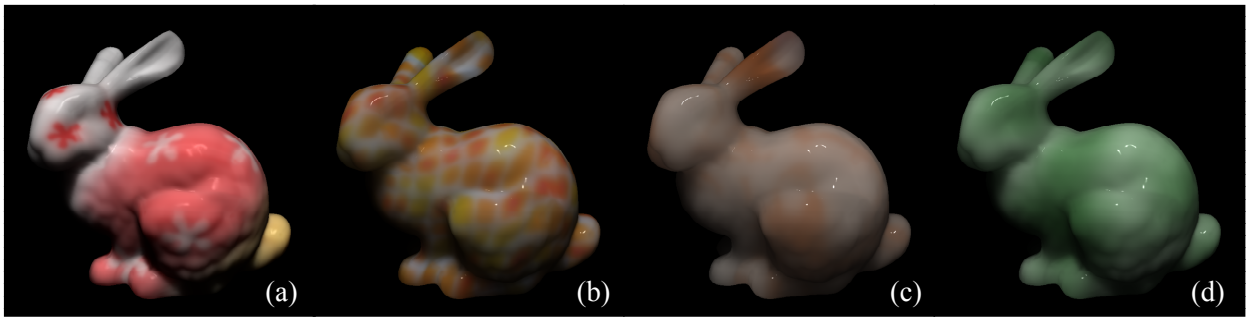

Fig. 19. Rendering results of a bunny model. (a) Constructed from an acquired wax material. (b-d) Edited with our system by a user.

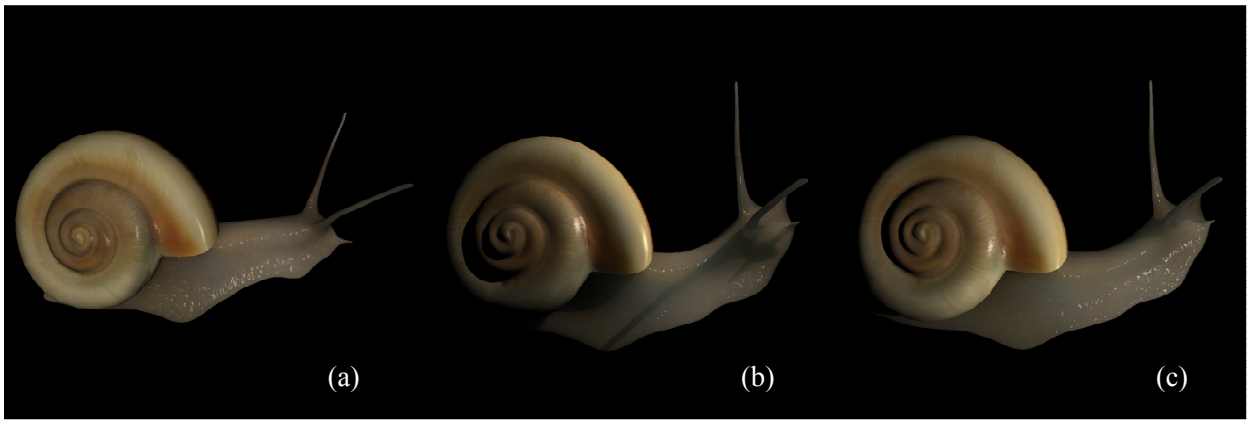

Fig. 20. The rendering result of a snail with local lighting. The translucent material of the snail body was designed using our editing tool, while the snail shell was modeled and rendered as an opaque surface.

Appendix A: Adjoint Method for the Inverse Diffusion Problem

In the following, we show the deduction of the adjoint method for $M = 1$. Generalization to larger values of M is straightforward.

The adjoint function is the Lagrangian multiplier $\varphi_1(x)$ in the following augmented objective function:

$$\tilde{f}_1(\vec{\mu},\vec{\kappa};\phi_1,\varphi_1) = f_1(\vec{\mu},\vec{\kappa}) - \int_V \varphi_1(x)\left\{\nabla\cdot[\kappa(x)\nabla\phi_1(x)] - \mu(x)\phi_1(x)\right\}\mathrm{d}V,$$

where $f_1(\vec{\mu},\vec{\kappa}) = \int_A \left(L_{o,1}(x) - L^R_{o,1}(x)\right)^2 \mathrm{d}A + \lambda\int_V \|\nabla\kappa\|^2\,\mathrm{d}V$. From the boundary condition of Eq. (2) and Eq. (3), we see that $L^R_{o,1}(x) = (\phi_1(x) - q_1(x))/2\pi$, where $x \in A$. The adjoint equation is the equation for $\varphi_1(x)$ such that $\partial\tilde{f}_1/\partial\phi_1 = 0$. By definition, the change in $\partial\tilde{f}/\partial\phi_1$ for a small perturbation $\delta\phi_1$ of ϕ_1 is the linear component of the difference $\tilde{f}_1(\vec{\mu},\vec{\kappa};\phi_1,\varphi_1) - \tilde{f}_1(\vec{\mu},\vec{\kappa};\phi_1 + \delta\phi_1,\varphi_1)$. By Green's formula and the boundary condition in Eq. (2) for ϕ_1, we have

$$\begin{aligned}
\tilde{f}_1\left(\bar{\mu},\bar{\kappa};\phi_1,\varphi_1\right) &= \int_A \left(L_{o,1}(x) - L^R_{o,1}(x)\right)^2 \mathrm{d}A + \lambda\int_V \|\nabla\kappa\|^2\,\mathrm{d}V \\
&\quad - \int_V \varphi_1(x)\left\{\nabla\cdot[\kappa(x)\nabla\phi_1(x)] - \mu(x)\phi_1(x)\right\}\mathrm{d}V \\
&= \int_A \left[L_{o,1}(x) - \frac{1}{2\pi}\left(\phi_1(x) - q_1(x)\right)\right]^2 \mathrm{d}A + \lambda\int_V \|\nabla\kappa\|^2\,\mathrm{d}V \\
&\quad - \int_V \phi_1(x)\left\{\nabla\cdot[\kappa(x)\nabla\varphi_1(x)] - \mu(x)\varphi_1(x)\right\}\mathrm{d}V \\
&\quad - \frac{1}{2C}\int_A \varphi_1(x)q_1(x)\mathrm{d}A + \int_A \left[\frac{1}{2C}\varphi_1(x) + \kappa(x)\frac{\partial\varphi_1(x)}{\partial n}\right]\phi_1(x)\mathrm{d}A.
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
&\tilde{f}_1\left(\bar{\mu},\bar{\kappa},\phi_1+\delta\phi_1;\varphi_1\right)-\tilde{f}_1\left(\bar{\mu},\bar{\kappa},\phi_1;\varphi_1\right)\\
=&-\frac{1}{\pi}\int_A \delta\phi_1\left[L_{o,1}(x)-\frac{1}{2\pi}\left(\phi_1(x)-q_1(x)\right)\right]\mathrm{d}A\\
&-\int_V \delta\phi_1(x)\left\{\nabla\cdot\left[\kappa(x)\nabla\varphi_1(x)\right]-\mu(x)\varphi_1(x)\right\}\mathrm{d}V\\
&+\int_A\left[\frac{1}{2C}\varphi_1(x)+\kappa(x)\frac{\partial\varphi_1(x)}{\partial n}\right]\delta\phi_1(x)\mathrm{d}A+O(\|\delta\phi_1\|^2).
\end{aligned}$$

As $\partial\tilde{f}_1/\partial\phi_1$ must be a zero operator, the linear component of the above perturbation must be zero for all possible $\delta\phi_1$. So the adjoint equation is

$$\begin{aligned}
&\nabla\cdot\left[\kappa(x)\nabla\varphi_1(x)\right]-\mu(x)\varphi_1(x)=0, && x\in V,\\
&\varphi_1(x)+2C\kappa(x)\frac{\partial\varphi_1(x)}{\partial n}=\frac{2C}{\pi}\left[L_{o,1}(x)-\frac{1}{2\pi}\left(\phi_1(x)-q_1(x)\right)\right], && x\in A.
\end{aligned}$$

And the gradient of the original function can be solved using the adjoint function as

$$\begin{aligned}
&\frac{\mathrm{d}f_1}{\mathrm{d}\kappa}=\frac{\partial\tilde{f}_1}{\partial\kappa}=\nabla\varphi_1(x)\cdot\nabla\phi_1(x)-2\lambda\Delta\kappa(x),\\
&\frac{\mathrm{d}f_1}{\mathrm{d}\mu}=\frac{\partial\tilde{f}_1}{\partial\mu}=\varphi_1(x)\phi_1(x),
\end{aligned}\qquad x\in V$$

with the boundary condition $\partial\kappa/\partial n=0$ for κ and $\partial\mu/\partial n=0$ for μ.

Appendix B: Discrete Partial Derivatives for Forward Diffusion

To compute the gradient and Laplacian of a funtion f on a node v_0 of the polygrid, we have

$$\begin{aligned}
f(v_j)-f(v_0)\approx&\frac{\partial f}{\partial x}\Delta x_j+\frac{\partial f}{\partial y}\Delta y_j+\frac{\partial f}{\partial z}\Delta z_j\\
&+\frac{1}{2}\cdot\frac{\partial^2 f}{\partial x^2}\Delta x_j^2+\frac{1}{2}\cdot\frac{\partial^2 f}{\partial y^2}\Delta y_j^2+\frac{1}{2}\cdot\frac{\partial^2 f}{\partial z^2}\Delta z_j^2\\
&+\frac{\partial^2 f}{\partial x\partial y}\Delta x_j\Delta y_j+\frac{\partial^2 f}{\partial y\partial z}\Delta y_j\Delta z_j+\frac{\partial^2 f}{\partial z\partial x}\Delta z_j\Delta x_j,
\end{aligned}\tag{10}$$

where $v_j=v_0+(\Delta x_j,\Delta y_j,\Delta z_j)$, $j=1,2,\ldots,6$ are the six nodes connected to v_0. For near-regular grids, if the coordinate directions are roughly along the directions of the grids, we may assume that

$$\begin{aligned}
&\text{for}j=1,2,\ |\Delta y_j|\ll|\Delta x_j|\text{ and }|\Delta z_j|\ll|\Delta x_j|;\\
&\text{for}j=3,4,\ |\Delta x_j|\ll|\Delta y_j|\text{ and }|\Delta z_j|\ll|\Delta y_j|;\\
&\text{for}j=5,6,\ |\Delta x_j|\ll|\Delta z_j|\text{ and }|\Delta y_j|\ll|\Delta z_j|.
\end{aligned}\tag{11}$$

Since $\frac{\partial^2 f}{\partial x\partial y}\Delta x_j\Delta y_j\approx 0$, $\frac{\partial^2 f}{\partial y\partial z}\Delta y_j\Delta z_j\approx 0$, and $\frac{\partial^2 f}{\partial z\partial x}\Delta z_j\Delta x_j\approx 0$, $\forall j$, we may discard the last three terms in (10). Then (10) can be written in matrix form:

$$\begin{pmatrix}
\Delta x_1 & \Delta x_1^2/2 & \Delta y_1 & \Delta y_1^2/2 & \Delta z_1 & \Delta z_1^2/2\\
\Delta x_2 & \Delta x_2^2/2 & \Delta y_2 & \Delta y_2^2/2 & \Delta z_2 & \Delta z_2^2/2\\
\Delta x_3 & \Delta x_3^2/2 & \Delta y_3 & \Delta y_3^2/2 & \Delta z_3 & \Delta z_3^2/2\\
\Delta x_4 & \Delta x_4^2/2 & \Delta y_4 & \Delta y_4^2/2 & \Delta z_4 & \Delta z_4^2/2\\
\Delta x_5 & \Delta x_5^2/2 & \Delta y_5 & \Delta y_5^2/2 & \Delta z_5 & \Delta z_5^2/2\\
\Delta x_6 & \Delta x_6^2/2 & \Delta y_6 & \Delta y_6^2/2 & \Delta z_6 & \Delta z_6^2/2
\end{pmatrix}
\begin{pmatrix}
\partial f/\partial x\\ \partial^2 f/\partial x^2\\ \partial f/\partial y\\ \partial^2 f/\partial y^2\\ \partial f/\partial z\\ \partial^2 f/\partial z^2
\end{pmatrix}=
\begin{pmatrix}
f(v_1)-f(v_0)\\ f(v_2)-f(v_0)\\ f(v_3)-f(v_0)\\ f(v_4)-f(v_0)\\ f(v_5)-f(v_0)\\ f(v_6)-f(v_0)
\end{pmatrix}.$$

Therefore,

$$\begin{pmatrix} \partial f/\partial x \\ \partial^2 f/\partial x^2 \\ \partial f/\partial y \\ \partial^2 f/\partial y^2 \\ \partial f/\partial z \\ \partial^2 f/\partial z^2 \end{pmatrix} = W \begin{pmatrix} f(v_1)-f(v_0) \\ f(v_2)-f(v_0) \\ f(v_3)-f(v_0) \\ f(v_4)-f(v_0) \\ f(v_5)-f(v_0) \\ f(v_6)-f(v_0) \end{pmatrix},$$

where

$$W = \begin{pmatrix} \Delta x_1 & \Delta x_1^2/2 & \Delta y_1 & \Delta y_1^2/2 & \Delta z_1 & \Delta z_1^2/2 \\ \Delta x_2 & \Delta x_2^2/2 & \Delta y_2 & \Delta y_2^2/2 & \Delta z_2 & \Delta z_2^2/2 \\ \Delta x_3 & \Delta x_3^2/2 & \Delta y_3 & \Delta y_3^2/2 & \Delta z_3 & \Delta z_3^2/2 \\ \Delta x_4 & \Delta x_4^2/2 & \Delta y_4 & \Delta y_4^2/2 & \Delta z_4 & \Delta z_4^2/2 \\ \Delta x_5 & \Delta x_5^2/2 & \Delta y_5 & \Delta y_5^2/2 & \Delta z_5 & \Delta z_5^2/2 \\ \Delta x_6 & \Delta x_6^2/2 & \Delta y_6 & \Delta y_6^2/2 & \Delta z_6 & \Delta z_6^2/2 \end{pmatrix}^{-1}.$$

If the chosen coordinate is not along the grid directions, we may choose a local coordinate frame (x', y', z'), such that the new coordinate directions are roughly along near-regular grid directions at v_0. Due to the rotational invariance of the Laplacian operator

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = \frac{\partial^2 f}{\partial x'^2} + \frac{\partial^2 f}{\partial y'^2} + \frac{\partial^2 f}{\partial z'^2},$$

we may compute the coefficients of $\partial^2 f/\partial x'^2$, $\partial^2 f/\partial y'^2$, $\partial^2 f/\partial z'^2$ as linear combinations of $f(v_j)-f(v_0)$ in the new local coordinate, and form the coefficient matrix for $f(v_j)-f(v_0)$ in the original coordinate.

In our application, we define $f = \kappa\phi$ as in [Stam 1995], and compute the coefficient matrix W at $v_i = v_0$ using a linear system solver. Then w_{ji} is computed as

$$w_{ji} = W_{2j} + W_{4j} + W_{6j}.$$

For the relationships in Eq. (11) to hold on an arbitrary object volume, a conformal mapping between the polygrid in the polycube and the object volume is needed. Although our mapping may not be exactly conformal, it leads to acceptable solutions.