Shell Texture Functions

Yanyun Chen Xin Tong Jiaping Wang[*] Stephen Lin Baining Guo Heung-Yeung Shum

Microsoft Research Asia

Abstract

We propose a texture function for realistic modeling and efficient rendering of materials that exhibit surface mesostructures, translucency and volumetric texture variations. The appearance of such complex materials for dynamic lighting and viewing directions is expensive to calculate and requires an impractical amount of storage to precompute. To handle this problem, our method models an object as a shell layer, formed by texture synthesis of a volumetric material sample, and a homogeneous inner core. To facilitate computation of surface radiance from the shell layer, we introduce the *shell texture function* (STF) which describes voxel irradiance fields based on precomputed fine-level light interactions such as shadowing by surface mesostructures and scattering of photons inside the object. Together with a diffusion approximation of homogeneous inner core radiance, the STF leads to fast and detailed raytraced renderings of complex materials.

Keywords: Texture mapping, reflectance and shading models, mesostructure, subsurface scattering, texture synthesis, BTF

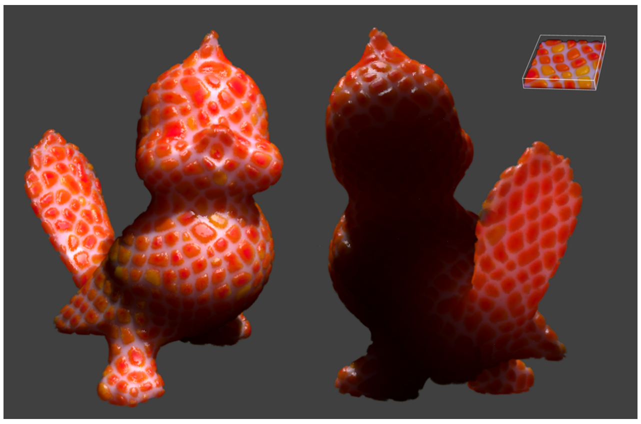

Figure 1: Shell texture function renderings of a non-homogeneous bird with surface mesostructures, from different viewing angles. Evident are detailed appearance features such as mesostructure silhouettes, translucency with material variations, and transmission of backlighting.

1 Introduction

The appearance of an object arises from a multitude of light interactions on and within the object volume. These various forms of reflection and scattering produce visual effects such as shadowing, masking, interreflection, translucency and fine-scale silhouettes which elevate the realism of rendered images when they are accounted for. These interactions are physically governed by shape and material attributes that typically vary over an object.

Texture functions have long been used to map spatially-variant material attributes such as color, surface normal perturbations [Blinn 1978], height field displacements [Cook 1984] and volumetric geometry [Neyret 1998] onto surfaces. While these methods can adequately model fine-scale surface geometry, known as *mesostructure*, and its interaction with illumination, appearance effects that arise from light transport within the material are not considered. Since many materials in the physical world are translucent to some degree, subsurface scattering is a vital element in realistic appearance [Hanrahan and Krueger 1993]. Subsurface scattering and other visual phenomena of a material can be captured in an image-based representation such as a *bidirectional texture function* (BTF) [Dana et al. 1999], which records the appearance of a surface patch under various lighting and viewing directions. The BTF, however, is an incomplete representation for mapping and synthesis, because it does not provide explicit shape information needed for rendering mesostructure silhouettes. Moreover, the effects of subsurface scattering are inadequately captured, e.g., backlighting through translucent objects is not captured in BTF acquisition.

[*]This work was done while Jiaping Wang was visiting Microsoft Research Asia from the Institute of Computing Technology, Chinese Academy of Sciences.

In this paper, we present a method that models and renders not only mesostructure shadowing, masking, interreflection and silhouettes on a surface, but also subsurface scattering within a non-homogeneous volume. For modeling, we propose an object representation consisting of a volumetric shell layer and an inner core, as illustrated in Fig. 2. The shell is created by texture synthesis using a volumetric material sample that can contain mesostructures and material non-homogeneities. Since material non-homogeneities from deep within the volume have a relatively subtle effect on appearance, the inner core is modelled as a homogeneous material.

Rendering this model with a full participating media simulation [Jensen and Christensen 1998] could capture all the aforementioned appearance details but also would incur a large computational expense. To efficiently render the detailed shell-core model, we propose the *shell texture function* (STF), which represents the irradiance distribution of a shell voxel with respect to incident illumination direction. The STF is precomputed by photon mapping that accounts for the fine-level lighting interactions among surface mesostructures and the scattering and absorption of photons within the possibly non-homogeneous material volume. With STF irradiance values for each shell voxel and a rapid simulation of subsurface scattering in the homogeneous inner core using the dipole diffusion approximation of [Jensen and Buhler 2002], objects with complex mesostructure and volume non-homogeneity can be efficiently rendered by simple radiance calculations and ray tracing.

Using the STF in the shell-core framework allows us to efficiently generate realistic images of complex materials such as shown in Fig. 1. We note that detailed silhouettes of the orange mesostructure protrusions cannot be rendered by mapping BTF images onto the mesh, and the transmissions of backlighting shown in the right-

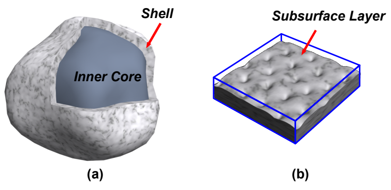

Figure 2: Shell-core object model. (a) An object is composed of a volumetric shell and an inner core. (b) The shell is formed by texture synthesis of a material sample. The sample is represented by a rectangular volume with mesostructure, such that it contains voxels of both the subsurface layer and the surrounding free space.

most image cannot be reasonably modelled in an image-based texture, since backlighting appearance captured from a given geometry is not fit for use on arbitrary shapes. As a texture function in a shell-core model, the STF of a volumetric material sample has the flexibility to be used with any mesh.

2 Related Work

Mesostructures and subsurface scattering are significant factors that determine the appearance of real-world surfaces. In previous work on subsurface scattering, detailed renderings of translucent objects have been computed by simulations of radiative transfer through a participating medium [Dorsey et al. 1999; Pharr and Hanrahan 2000]. Although these simulations can handle materials of arbitrary composition, they are computationally expensive, and other methods achieve greater efficiency by specifically addressing translucent materials that are homogeneous. Hanrahan and Krueger [Hanrahan and Krueger 1993] presented an analytic expression for single scattering in a homogeneous, uniformly lit slab. To this single scattering solution, Jensen et al. used diffusion approximations for multiple scattering to rapidly compute subsurface scattering in homogeneous media [Jensen et al. 2001; Jensen and Buhler 2002] (see also [Koenderink and van Doorn 2001]). Recently, several researchers have proposed methods for interactive rendering of homogeneous scattering materials [Mertens et al. 2003; Hao et al. 2003]. A large number of translucent objects, however, are not homogeneous, such as veined marble and human skin, and an efficient rendering technique that accommodates some degree of non-homogeneity is desirable. Moreover, previous subsurface methods do not account for surface mesostructures, which affect the distribution of light entering a volume.

Koenderink and van Doorn pointed out the importance of mesostructures to surface appearance, and noted the shortcomings of traditional texture mapping and bump mapping for mesostructure rendering [Koenderink and Doorn 1996]. Dana et al. introduced the BTF as an image-based description of mesostructures, and measured the BTFs of many real-world surfaces to form the CUReT database [Dana et al. 1999]. Leung and Malik introduced 3D textons for analyzing surface mesostructures [Leung and Malik 2001], and building on this work, Tong et al. derived an algorithm for synthesizing a BTF on arbitrary surfaces [Tong et al. 2002].

Besides BTFs, other image-based representations of appearance have been proposed, such as surface light fields [Wood et al. 2000; Chen et al. 2002] and polynomial texture maps [Malzbender et al. 2001]. These image-based descriptors are intended for restricted lighting and viewing conditions. Surface light fields and view-dependent texture maps record surface appearance from varying viewing directions under a fixed lighting condition, while polynomial texture maps represent appearance for fixed viewing geometry and variable illumination. Additionally, image-based techniques [Wood et al. 2000; Chen et al. 2002; Debevec et al. 2000; Matusik et al. 2002] model the appearance of a given object. In all of these image-based methods, translucency features such as transmission of backlighting are represented specifically for the geometry of the captured object and are generally unsuitable for mapping onto arbitrary shapes. Our work addresses efficient rendering of complex materials on arbitrary meshes under dynamic lighting and viewing conditions, which requires texture functions with more modeling flexibility and generality than that practically provided by image-based structures.

Objects with intricate surface geometry can be rendered by volume texturing techniques [Neyret 1998; Lengyel et al. 2001], which map a replicated volumetric pattern onto a surface. Volume texture variations can also be formed by procedural modeling [Ebert et al. 1994], which is effective for certain classes of materials (wood, marble, etc.). These methods concentrate on efficient handling of fine geometric or material detail and do not address the complicated reflectance that results. In [Dorsey et al. 1999], a method for modeling the detailed appearance of weathered stone is presented, using an inhomogeneous volume layer around an opaque core. This work presents a technique specific to comprehensive modeling of stone weathering, for which rendering is done with an expensive full participating media simulation. Our work develops a general-purpose texture function for complex materials, and addresses efficient rendering that accounts for detailed light interactions on and within such materials.

3 Overview

Our proposed method models objects as two components: a heterogeneous shell layer with mesostructures and a homogeneous inner core. The basic idea behind the inner core is that light from deep within a volume undergoes significant scattering such that its radiance exhibits an averaging effect of its material properties. To take advantage of this characteristic, the inner core is modelled as a homogeneous material, for which radiance can be rapidly computed [Jensen and Buhler 2002]. In contrast, shell voxels lie near the object surface, so their variations in material properties have a relatively large effect on appearance.

Since the combination of volume non-homogeneities with surface mesostructures results in complicated scattering of light within the shell, it is desirable to precompute this scattering as much as possible to facilitate run-time rendering. The quantities we precompute are the STF voxel irradiances in the volumetric material sample with respect to illumination direction, as described in Section 4. The STF allows rapid determination of radiance for each voxel in the sample volume by simple calculations. Since the shell is synthesized from the volumetric material sample onto a closed mesh, the radiance of the shell voxels and at the shell surface can be quickly computed from the STF as well.

Based on the STF, our modeling and rendering technique performs the following steps:

- Acquisition of material parameters for voxels in the volumetric material sample, described in Section 5.1.
- STF construction by photon mapping in the material sample, presented in Section 5.2.
- Object shell formation by texture synthesis of the material sample onto a given mesh, outlined in Section 6. With shell

synthesis, STF values are effectively assigned to each shell voxel.

- Rendering with the STF and the dipole approximation of the inner core, explained in Section 7.

Some results of our algorithm and a discussion of our method are presented in Section 8, followed by conclusions and future work in Section 9.

4 The STF Representation

The STF represents irradiance distributions of points in a material sample for all incident illumination directions. The material sample is modelled as a volume V_b called the STF base volume. V_b contains an $n_x \times n_y \times n_z$ array of voxels on a regular 3D grid, where $n_x \times n_y$ determines the size of the texture parameter space and n_z defines the thickness of the STF. Since mesostructure is represented in the base volume, the voxels of V_b may lie in either the subsurface layer or the surrounding free space, as illustrated in Fig. 2(b). For each subsurface voxel $\mathbf{x}$, we store the following material properties: extinction coefficient $\kappa(\mathbf{x})$, albedo $\alpha(\mathbf{x})$, and phase function $f(\mathbf{x},\omega',\omega)$, where ω and ω' are the incoming and outgoing light directions, respectively. The scattering coefficient σ_s of a material is related to extinction and albedo as $\sigma_s = \alpha\kappa$, and the absorption coefficient is defined as $\sigma_a = \kappa - \sigma_s$. The extinction coefficient $\kappa(\mathbf{x})$ and albedo $\alpha(\mathbf{x})$ describe the radiative transfer properties of a participating medium [Siegel and Howell 1992], which determine the translucency and chromatic properties of a voxel. A flag for indicating surface voxels is also included, and when it is on, the surface normal and relative index of refraction are stored as well.

Now we define the 5D STF by specifying its single- and multiple-scattering components $I_s(\mathbf{x},\omega_l)$ and $I_m(\mathbf{x},\omega_l)$, where $\mathbf{x}$ is the position in V_b and ω_l is the light direction. We also describe how these STF values allow rapid computation of the radiance $L(\mathbf{x},\omega)$ at any point $\mathbf{x}$ of V_b and in any direction ω. According to the light transport equation in a participating medium [Siegel and Howell 1992], the radiance $L(\mathbf{x},\omega)$ is expressed as

$$(\omega \cdot \nabla)L(\mathbf{x},\omega) = \sigma_a(\mathbf{x})L_e(\mathbf{x},\omega) + \sigma_s(\mathbf{x})L_i(\mathbf{x},\omega) - \kappa(\mathbf{x})L(\mathbf{x},\omega). \tag{1}$$

$L_e(\mathbf{x},\omega)$ is the emitted radiance, and $L_i(\mathbf{x},\omega)$ is the in-scattered radiance. An efficient way to evaluate the radiance $L(\mathbf{x},\omega)$ is using the ray marching algorithm [Ebert et al. 1994]. By far, the most computationally expensive part of the ray march is the calculation of in-scattered radiance, which can be computed using volume photon mapping [Jensen and Christensen 1998] as

$$L_i(\mathbf{x},\omega) = \frac{1}{\sigma_s(\mathbf{x})} \sum_{p=1}^{n} f(\mathbf{x},\omega_p,\omega) \frac{\Delta\Phi_p(\mathbf{x},\omega_p)}{\Delta V},$$

where the radiance is summed over the n photons inside a differential volume ΔV, and $\Delta\Phi_p(\mathbf{x},\omega_p)$ is the flux carried by a photon from incoming direction ω_p.

The flux is divided into single-scattering and multiple-scattering terms. This classification was used in [Jensen and Buhler 2002] because single scattering can be efficiently rendered by ray tracing, while multiple scattering through a homogeneous medium can be rapidly simulated using a dipole diffusion approximation. In our scenario, the dipole approximation cannot be employed since the material sample is non-homogeneous, but we take advantage of the property that multiple scattering can be considered approximately isotropic [Stam 1995]. For isotropic multiple scattering, we can re-write the expression of $L_i(\mathbf{x},\omega)$ as

$$f(\mathbf{x},\omega_l,\omega)\frac{\sum_s \Delta\Phi_p(\mathbf{x},\omega_l)}{\sigma_s(\mathbf{x})\Delta V} + \frac{1}{4\pi}\frac{\sum_m \Delta\Phi_p(\mathbf{x},\omega_p)}{\sigma_s(\mathbf{x})\Delta V},$$

where $\sum_s$ and $\sum_m$ are sums over single- and multiple-scattering photons respectively (we include a photon in $\sum_m$ only if it is actually scattered). For single-scattering photons, their incoming directions must be the light direction ω_l. Although multiple scattering is assumed to be isotropic, it is still a function of light direction ω_l because of surface mesostructures, which affect the direct illumination arriving at each surface point. The in-scattered radiance is finally formulated as

$$L_i(\mathbf{x},\omega) = f(\mathbf{x},\omega_l,\omega)I_s(\mathbf{x},\omega_l) + \frac{1}{4\pi}I_m(\mathbf{x},\omega_l)$$

where

$$I_s(\mathbf{x},\omega_l) = \frac{\sum_s \Delta\Phi_p(\mathbf{x},\omega_l)}{\sigma_s(\mathbf{x})\Delta V},\ I_m(\mathbf{x},\omega_l) = \frac{\sum_m \Delta\Phi_p(\mathbf{x},\omega_p)}{\sigma_s(\mathbf{x})\Delta V} \tag{2}$$

The quantities $I_s(\mathbf{x},\omega_l)$ and $I_m(\mathbf{x},\omega_l)$ represent the *voxel irradiance* at $\mathbf{x}$ when the incident lighting comes from direction ω_l. These single- and multiple-scattering irradiances are precomputed as the STF, a 5D function that can be easily stored. With the STF, $L_i(\mathbf{x},\omega)$ can be quickly calculated and Eq. (1) can be easily integrated through the material sample to compute radiance $L(\mathbf{x},\omega)$.

In theory, one could precompute or capture the reflectance field [Debevec et al. 2000] of a material sample to minimize run-time computation. The reflectance field, however, is an 8D function of a 4D field of incident illumination and a 4D field of radiance, hence too large to store. By considering only directional lighting, illumination becomes 2D and the reflectance field reduces to 6D, but mesostructure geometry still needs to be known to accurately render silhouettes. For practical reasons, we instead precompute and store the STF, an expensive 5D sub-function in the radiance calculation, and then perform the remaining relatively inexpensive computation at run time. Although precomputing a reflectance field for use with recorded mesostructure geometry can reduce run-time computation, the amount of additional data for the extra viewing dimensions, which should be sampled densely, can be quite substantial[1]. Furthermore, the STF combines naturally with volumetric mesostructures, while it is not clear how to combine a reflectance field with such (non-height-field) mesostructures. Because of these practical benefits of having a lower-dimensional texture function and since radiance from surface voxels can be calculated rather simply from irradiance data, we use precomputed voxel irradiances in our STF representation.

5 Construction of the STF Sample

To precompute the irradiance functions that compose the 5D STF, the surface mesostructures and scattering characteristics within the volumetric material sample V_b must first be determined. Although this could be done procedurally [Ebert et al. 1994], greater realism could be obtained by scanning material properties from real volumes [Hanrahan and Krueger 1993] and measuring mesostructures from real-world surfaces [Dana et al. 1999; Tong et al. 2002]. From the base volume, the STF is constructed by calculating and storing the single-scattering and multiple-scattering irradiance of each voxel under sampled light directions.

[1]For the strawberry example exhibited in Fig. 16, which has a non height field mesostructure, its reflectance field data would require 1.3 GB of memory, while the STF occupies a more manageable 159 MB of space, which is also more feasible for future hardware implementation.

5.1 Acquisition of the Base Volume

We generate the base volume V_b using volume modeling and processing techniques that are well-studied for volume visualization [Kaufman 1991]. There are two ways to obtain V_b. One is to measure real materials using volume imaging technologies such as computed tomography (CT), and the other is to scan convert existing 3D geometry models [Kaufman 1991].

For measured volume data, we create V_b using voxel classification followed by table lookups. Voxel classification is a standard process for determining material(s) contained in each voxel of measured volume data [Kaufman 1991]. The classification technique proposed in [Levoy 1988] can be used to determine voxel materials in certain classes of volumes, and for each material its extinction coefficient κ and albedo α can be obtained in published measurement tables such as the one in [Jensen et al. 2001]. Extinction coefficients and albedos of multiple materials in the same voxel are averaged according to their relative composition in the voxel. Voxels that lie on the mesostructure surface are identified as those that lie along the outermost isosurface [Kaufman 1991]. In the STF base volume, a binary function is used to indicate voxels on the mesostructure surface.

In scan conversion, 3D geometry models are converted into volume data [Kaufman 1991]. A 3D geometry model usually consists of several components, each associated with a given material. When a geometric model is scan converted, the conversion algorithm determines the material(s) in each voxel, and as in the case of measured volume data, the voxel extinction coefficient and albedo can be determined from tables. As in previous work [Jensen et al. 2001; Jensen and Buhler 2002] we use the Henyey-Greenstein phase function [Henyey and Greenstein 1941] for both measured and scan converted volume data.

For some materials such as veined marble, a spatially-varying albedo map $\alpha(\mathbf{x})$ is needed, but published measurement tables generally provide only a single albedo value [Jensen et al. 2001]. To obtain an albedo map we first generate voxel colors for the object interior using solid texture synthesis, which can be either procedural [Ebert et al. 1994], sample-based [Wei 2001], or sample-view-based [Dischler et al. 1998; Wei 2001]. The procedural approach works for limited types of textures. The sample-based approach works for any solid texture in principle, but it requires as input a solid texture sample, which may not be easy to obtain. The sample-view-based approach requires only a few (typically $2 \sim 3$) 2D images of the target solid texture and is thus easier to use. The voxel colors can then be converted to albedos $\alpha(\mathbf{x})$ using Jensen's technique [Jensen and Buhler 2002] by treating voxel colors as diffuse reflectance coefficients.

The scattering properties of the homogeneous inner core are determined from properties of the core surface, according to the dipole diffusion approximation (with surface texture extension) [Jensen et al. 2001]. In our shell-core model, the core surface properties are those of the bottom layer of voxels in the shell.

5.2 STF Irradiance Sampling

Using the material properties of V_b, the STF irradiance values for each voxel $\mathbf{x}$ of V_b are computed for sampled illumination directions, as illustrated in Fig. 3(a). For a given light direction ω_l, we compute $I_s(\mathbf{x},\omega_l)$ and $I_m(\mathbf{x},\omega_l)$ using a variant of the algorithm proposed in [Jensen and Christensen 1998] for photon tracing in a participating medium. We emit photons along a light direction ω_l and evaluate their contributions to the STF as photons reflect off of the

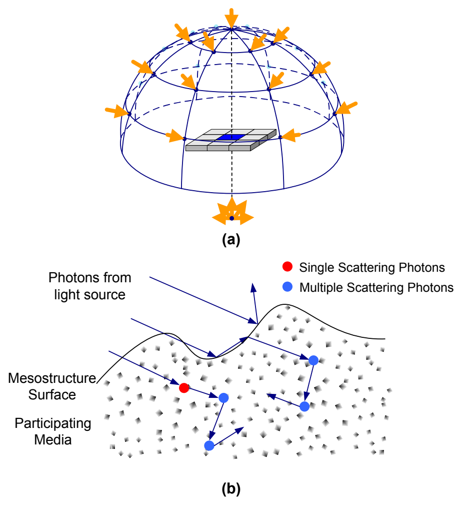
(a)
Photons from light source
● Single Scattering Photons
● Multiple Scattering Photons
Mesostructure Surface
Participating Media
(b)

Figure 3: STF irradiance sampling. (a) To sample irradiance within an STF base volume, we surround it by eight other STF base volumes and calculate irradiance data for sampled lighting directions (marked by orange arrows). (b) Photon tracing in the STF base volume.

mesostructure surface and scatter within the object interior. From the mapped photons, we evaluate the STF components of a voxel at $\mathbf{x}$ according to Eq. (2). In our implementation, ΔV is chosen to be a sphere with a radius of one voxel.

To avoid boundary effects in STF sampling, we surround the base volume by eight other identical volumes as shown in Fig. 3(a). Here we assume that the base volume is tilable and has texture dimensions n_x, n_y greater than twice the diffuse mean free path of V_b [Jensen and Buhler 2002]. Since V_b can be non-homogeneous, its diffuse mean free path is computed as a weighted average among the materials that compose V_b. This constraint on V_b dimensions ensures that the eight identical volumes around V_b provide a sufficiently large volume neighborhood for computing the STF of each voxel in V_b. The thickness n_z of V_b is set to be greater than the maximum mesostructure depth plus three times the average mean free path of V_b. This choice of n_z ensures that we can ignore the single-scattering contribution from the homogeneous inner core. A non-tilable base volume of insufficient size can be converted to a larger tilable volume using constrained solid texture synthesis [Wei 2001].

The photon tracing process is illustrated in Fig. 3(b). A photon p first interacts with the mesostructure surface where it is either reflected at the surface or refracted into the object interior, according to Fresnel's formulae for unpolarized light. Photons that reflect off a surface point are not recorded, because these interactions are handled by raytracing during the rendering stage. A photon that propagates through the object interior can either pass through the medium unaffected or interact with it. Each time the photon p interacts with the medium at a voxel $\mathbf{x}$, its contribution is added to $I_s(\mathbf{x},\omega_l)$ if it is p's first interaction or to $I_m(\mathbf{x},\omega_l)$ otherwise, where surface reflection is considered an interaction. The probability that the photon p interacts with the medium at position $\mathbf{x}$ is determined by the following cumulative probability density function

$$p(\mathbf{x}) = 1 - e^{-\int_{\mathbf{x}_0}^{\mathbf{x}} \kappa(\xi) d\xi}$$

where $\mathbf{x}_0$ is the mesostructure surface point where p refracted into the object interior. The integral in the equation is evaluated by the ray marching method described in [Dorsey et al. 1999]. If a pho-

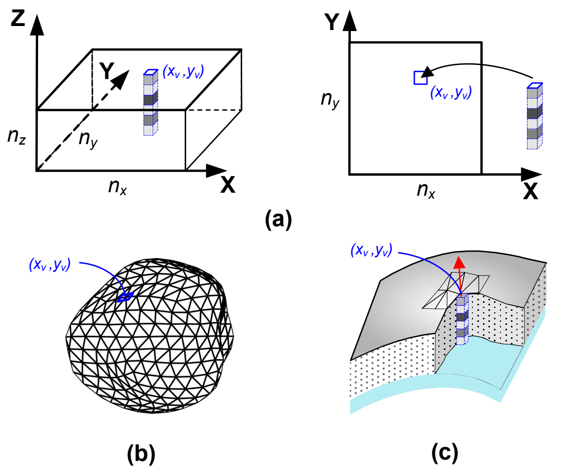

Figure 4: Object shell synthesis. (a) The STF base volume is regarded as a 2D texture $T(x,y)$, in which each $T(x,y)$ is a stack of voxels. (b) For each mesh vertex $\mathbf{v}$, the synthesis algorithm assign it a texture coordinate (x_v, y_v). (c) The stack of voxels at (x_v, y_v) in the texture domain is placed in the object shell right below vertex $\mathbf{v}$ along the surface normal at $\mathbf{v}$.

ton interacts with the medium, Russian roulette decides whether the photon is scattered or absorbed according to a scattering probability based on the albedo $\alpha(\mathbf{x})$. The direction of the scattered photon is calculated by importance sampling of the phase function $f(\mathbf{x}, \omega', \omega)$.

We perform photon tracing for a set of discretized light directions which sample the upper and lower hemispheres. For each light direction ω_l, a large number of photons are traced and their contributions to the STF are added to $I_s(\mathbf{x}, \omega_l)$ and $I_m(\mathbf{x}, \omega_l)$. In our implementation, 1000 photons are traced for each RGB channel. The upper hemisphere is sampled by $72 = 6 \times 12$ light directions. The lower hemisphere is sampled identically, but this incident light is considered to be isotropic multiple scattering, because backlighting arrives from the homogeneous inner core. Based on this property, the multiple-scattering STF values of all directions in the lower hemisphere are averaged and recorded as $I_m(\mathbf{x}, \omega_b)$ where ω_b represents all backlight directions.

A moderate amount of storage is needed for the STF. For example, the material sample for the strawberry shown in Fig. 16, which has a $96 \times 96 \times 10$ base volume and is sampled under 73 light directions, occupies 205 MB of storage space before compression. For compression, we employ a scheme similar to [Beers et al. 1996] using vector quantization (VQ) followed by entropy coding, where each VQ codeword is a 6D vector of the single- and multiple-scattering irradiance colors for a given voxel and lighting direction. For STFs, this scheme can yield a compression ratio of 48:1.

6 Shell Synthesis

To form the shell model, we synthesize the material base volume V_b onto the target mesh, such that each point in the shell is assigned a texture coordinate in the base volume V_b. Each shell voxel is then assigned the STF values of the base volume voxel with which it is associated.

Object Shell Synthesis: Fig. 4 illustrates the synthesis of the material shell from the $n_x \times n_y \times n_z$ STF base volume of the material sample. For this synthesis we regard V_b as a 2D texture $T(x,y)$ of $n_x \times n_y$ texels. Each texel $T(x_i, y_j)$ consists of a stack of n_z voxels $\{(x_i, y_j, z_1), \ldots, (x_i, y_j, z_{n_z})\}$ with their material properties. We synthesize the texture $T(x,y)$ onto the target surface mesh using the algorithm proposed by Tong et al. [Tong et al. 2002].

The basic approach of [Tong et al. 2002] is similar to several techniques for synthesizing color textures on surfaces [Turk 2001; Wei 2001; Ying et al. 2001] except that it can handle high-dimensional texture functions, i.e., 2D textures in which texels are high-dimensional vectors. This method is based on the observation that at a local scale a given high-dimensional texture sample contains only a small number of perceptually distinguishable mesostructures and reflectance variations, known as textons. The textons of a high-dimensional texture function are determined through a texton analysis or a K-means clustering step [Leung and Malik 2001], and then are encoded in a 2D texton map that captures the essence of the texture in a compact representation. Synthesis of this 2D texton map onto the target mesh effectively determines a mapping of the high-dimensional texture. The computational cost is similar to that of color texture synthesis (e.g. [Turk 2001; Wei 2001]) and can be accelerated using the k-coherence search technique [Tong et al. 2002].

For our material sample, synthesis is based on the following voxel properties: extinction coefficient κ, albedo α, and phase function $f(\mathbf{x}, \omega', \omega)$. Phase functions present a technical difficulty for our synthesis algorithm. Since they are a function of ω' and ω, the similarity between two phase functions cannot efficiently be measured. To reduce this problem, we follow the convention (e.g., [Jensen et al. 2001]) in which it is assumed that $f(\mathbf{x}, \omega', \omega) = f(\mathbf{x}, \omega' \cdot \omega)$, i.e., f depends only on $\mathbf{x}$ and the phase angle $\omega' \cdot \omega$. Under this assumption, we can measure the similarity of two phase functions $f(\mathbf{x}, \omega' \cdot \omega)$ and $f(\mathbf{x}', \omega' \cdot \omega)$ by densely sampling the phase angles at points $\mathbf{x}$ and $\mathbf{x}'$. For further simplification we employ the moments similarity relation [Wyman et al. 1989] which allows us to alter the scattering properties of the medium without significantly affecting the light distribution. With this, the scattering coefficient $\sigma_s(\mathbf{x})$ is reduced to

$$\sigma_s'(\mathbf{x}) = \sigma_s(\mathbf{x})(1 - \int_{4\pi} f(\mathbf{x}, \omega \cdot \omega')(\omega \cdot \omega')d\omega').$$

as done in [Jensen and Buhler 2002]. This moments similarity relation is an effective approximation when light scattering is strongly-peaked forward scattering [Wyman et al. 1989], which is the case for most translucent materials of interest in computer graphics [Jensen and Buhler 2002]. With the above simplifications, synthesis of the material shell is performed using the reduced extinction coefficient $\kappa'(\mathbf{x}) = \sigma_s'(\mathbf{x}) + \sigma_a(\mathbf{x})$ and albedo $\alpha'(\mathbf{x}) = \sigma_s'(\mathbf{x})/\kappa'(\mathbf{x})$. We note that the scattering coefficient is reduced only for texture synthesis purposes, and the original scattering coefficient is used to compute the STF.

The synthesis process assigns a 2D texture coordinate (x_v, y_v) to each vertex $\mathbf{v}$ on the target mesh. The material shell is then obtained as shown in Fig. 4 by placing n_z voxels $\{(x_v, y_v, z_1), \ldots, (x_v, y_v, z_{n_z})\}$ along the surface normal at every vertex $\mathbf{v}$. The material shell represented this way is a dense set of points on an irregular grid. To facilitate the subsequent rendering operations, these points are resampled onto a regular grid.

Shell Resampling: We generate a distance field of the target surface mesh on a regular grid, with the grid interval d_0 equal to the voxel size of the base volume [Payne and Toga 1992]. As shown in Fig. 5(a), for each sampling point $\mathbf{x}$ on the regular grid, we know the nearest point $\mathbf{v}_f$ on the mesh and the distance d between $\mathbf{v}_f$ and $\mathbf{x}$. The texture coordinate of $\mathbf{x}$ is then $(t_u, t_v, d/d_0)$, where (t_u, t_v) is the texture coordinate of the mesh vertex $\mathbf{v}$ nearest to $\mathbf{v}_f$. Using the

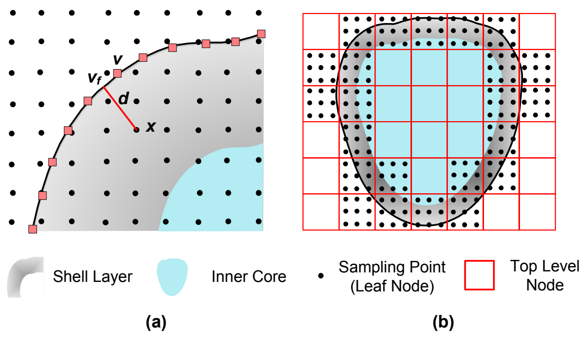

Figure 5: Shell resampling (2D case). (a) Computing the texture coordinates of point **x**. (b) Two-level storage structure for the shell voxels.

texture coordinate of the nearest vertex avoids interpolation calculations and provides a close approximation, because for synthesis the mesh is densely re-tiled such that each voxel corresponds to a vertex on the surface.

Since the material shell occupies only the space near the target surface mesh, many sampling points of the regular grid have no meaningful data to store. This allows us to achieve space efficiency by using hierarchical spatial data structures such as octrees and k-d trees. In our implementation, we use a simple two-level data structure shown in Fig. 5(b). The (bottom-level) leaf nodes correspond to the sampling points of the regular grid and store data for the subsurface layer. A top-level node contains $8 \times 8 \times 8$ leaf nodes. Space efficiency is achieved by setting top-level nodes to null in areas not occupied by the material shell.

Assigning STF Values: For a sampling point **x** in the material shell with texture coordinate $(t_u, t_v, d/d_0)$, its material properties are those of the corresponding point $\mathbf{x}' = (t_u, t_v, d/d_0)$ in the base volume. We further use the single- and multiple-scattering STF values at $\mathbf{x}'$ as those at **x**. This assignment of STF values is legitimate because in synthesis of the material shell, each point **x** has a volume neighborhood $N_\mathbf{x}$ whose material properties are similar to that of the volume neighborhood $N_{\mathbf{x}'}$ around point $\mathbf{x}'$ in the base volume V_b. For an optically thick medium and a given light direction, the multiple-scattering irradiance contribution to a point is primarily determined by the material properties of a sufficiently large volume neighborhood around the point; contributions from areas outside this neighborhood are negligible by comparison. For single scattering, photons from outside the neighborhood can bring irradiance to a voxel, but such photons are so few that their irradiance contribution can be safely ignored.

The validity of STF values assigned to shell voxels is contingent on the neighborhoods $N_\mathbf{x}, N_{\mathbf{x}'}$ being large enough to contain the voxels that affect the irradiance of $\mathbf{x}, \mathbf{x}'$. The range over which voxels affect each other is dependent on the material properties. As described in Section 4.2, we express this range as twice the diffuse mean free path of the material. This neighborhood size is included as a parameter in the texture synthesis algorithm to ensure that an appropriate range of neighborhood similarity is used.

The similarity of neighborhoods $N_\mathbf{x}, N_{\mathbf{x}'}$ can also be affected by curvature in the shell, which warps the configuration of $N_\mathbf{x}$. In this work, we assume the size of $N_\mathbf{x}$ to be relatively small in relation to variations in mesh shape, such that $N_\mathbf{x}$ is approximately flat. To roughly handle instances where this assumption does not hold, the STF could potentially be extended to include a single curvature parameter without requiring an impractical amount of storage.

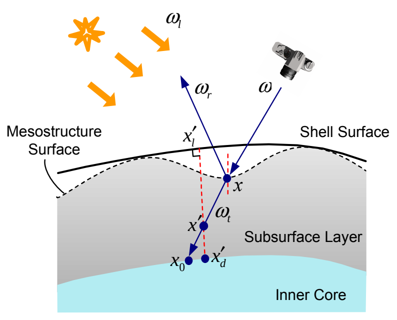

Figure 6: Ray tracing geometry.

7 STF Rendering

Represented by a STF-based shell and a homogeneous inner core, an object can be rendered efficiently by ray tracing. The radiance calculation involves three steps. First, the illumination that enters the homogeneous core is obtained as the irradiances at the bottom layer of shell voxels, calculated by multiplying the light intensity with the bottom layer STF values. Second, from the light entering the core, the dipole approximation is employed to compute the light exiting the homogeneous core. Third, the light incident upon the object and the light exiting the inner core are used to determine the irradiances of the shell voxels using the STF. From these irradiance values, the exiting radiance at the mesostructure surface is calculated by ray marching [Ebert et al. 1994] into the object interior.

Fig. 6 illustrates the geometry of radiance evaluation for a point or directional light source with local light direction ω_l. The radiance $L(\mathbf{x}, \omega)$ of a surface voxel **x** towards viewing direction ω is composed of radiance reflected off the object surface, L_R, and radiance exiting the object volume, L_T:

$$L(\mathbf{x}, \omega) = (1 - F_t) L_R(\mathbf{x}, \omega_r) + F_t L_T(\mathbf{x}, \omega_t) \tag{3}$$

where F_t is the Fresnel transmittance, ω_r is the mirror direction of ω across the mesostructure surface normal at **x**, and ω_t is the refracted ray direction with respect to ω. The value of $L_R(\mathbf{x}, \omega_r)$ is evaluated by conventional ray tracing, recursively spawning rays. Because of the recursive spawning, interreflections over the mesostructure surface are included in $L_R(\mathbf{x}, \omega_r)$.

To evaluate the refracted ray radiance $L_T(\mathbf{x}, \omega_t)$, we integrate along the refraction ray into the object interior according to the equation:

$$\int_{x_0}^{x} \sigma_s(\mathbf{x}') e^{-\int_{x'}^{x} \kappa(\xi) d\xi} I_b(\mathbf{x}', \omega_t) d\mathbf{x}',$$

where $I_b(\mathbf{x}', \omega_t)$ is the irradiance of voxel $\mathbf{x}'$ that contributes to radiance in direction ω_t. The irradiance distribution $I_b(\mathbf{x}', \omega_t)$ is computed from the local light source intensity $I_0(\mathbf{x}'_l, \omega_l)$ on the shell surface, the homogeneous core radiance $L_d(\mathbf{x}, \omega_l)$, and the single- and multiple-scattering STF values as follows:

$$\frac{1}{4\pi} I_0(\mathbf{x}'_l, \omega_l) I_m(\mathbf{x}', \omega_l) + \frac{1}{4\pi} L_d(\mathbf{x}'_d, \omega_l) I_m(\mathbf{x}', \omega_b) +$$
$$f(\mathbf{x}', \omega_l, \omega_t) I_0(\mathbf{x}'_l, \omega_l) I_s(\mathbf{x}', \omega_l). \tag{4}$$

The first term of this expression is the multiple-scattering STF value scaled by the illumination I_0 arriving at the top of the object shell. The second term accounts for the contribution of inner core radiance $L_d(\mathbf{x}'_d, \omega_l)$ from point $\mathbf{x}'_d$, the projection of $\mathbf{x}'$ in the shell normal direction onto the inner core. The value of $L_d(\mathbf{x}'_d, \omega_l)$ is obtained from the dipole calculation, and its contributions to shell

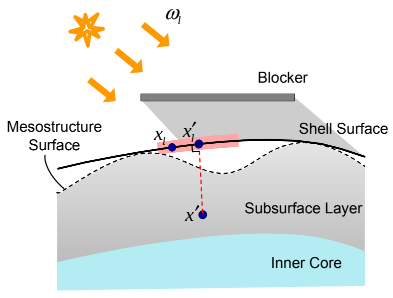

Figure 7: To handle instances where illumination onto the shell is not locally uniform, the irradiance correction kernel propagates shadow/light into the subsurface layer to account for the lighting differences.

voxel irradiances are scaled by the multiple-scattering STF values $I_m(\mathbf{x}', \omega_b)$ for backlight direction ω_b. The first two terms of $I_b(\mathbf{x}', \omega_t)$ together account for the isotropic multiple scattering contribution from voxels within the object shell. Finally, the third term of $I_b(\mathbf{x}', \omega_t)$ is the single scattering contribution from within the shell that is reflected in direction ω_t. We do not include the single scattering contribution of the homogeneous inner core because of the negligible likelihood of a photon traversing the subsurface layer without being scattered or absorbed.

For each sample point $\mathbf{x}$, we obtain $I_s(\mathbf{x}, \omega_l)$ and $I_m(\mathbf{x}, \omega_l)$ by trilinearly interpolating the corresponding single and multiple STF values of the eight surrounding voxels. Material properties needed for evaluating $L_T(\mathbf{x}, \omega_t)$ are also obtained by trilinear interpolation.

Irradiance Correction: Since the STF values $I_m(\mathbf{x}', \omega_l)$ and $I_s(\mathbf{x}', \omega_l)$ in Eq. (4) are precomputed with uniform directional illumination over the material sample, the calculation of $I_b(\mathbf{x}', \omega_t)$ for a shell voxel is valid only when the illumination is uniform over the surface neighborhood around $\mathbf{x}'_l$, the projection of $\mathbf{x}'$ onto the shell surface along the shell normal direction. While this lighting uniformity approximately holds for most points on the shell surface, abrupt illumination changes occur near boundaries of shadows cast by external geometry, as illustrated in Fig. 7. Unlike mesostructure shadows, these external geometry shadows are not modelled in the STF. For shell voxels near the shadow boundaries cast by external geometry, we employ an irradiance correction technique to improve the accuracy of irradiance calculation.

Let us first consider instances where $\mathbf{x}'$ is in shadow near a shadow boundary. In this case, $I_b(\mathbf{x}', \omega_t)$ is calculated based on the assumption that all shell surface voxels near $\mathbf{x}'_l$ have the same illumination $I_0(\mathbf{x}'_l, \omega_l)$. For a shell surface voxel $\mathbf{x}_l$ that is near $\mathbf{x}'_l$ but not in shadow, the difference between assumed and actual illumination at $\mathbf{x}_l$, $\Delta I_0(\mathbf{x}_l, \omega_l) = I_0(\mathbf{x}_l, \omega_l) - I_0(\mathbf{x}'_l, \omega_l)$, will be significant because of the abrupt change of illumination across a shadow boundary, and the calculation of irradiance $I_b(\mathbf{x}', \omega_t)$ must be corrected accordingly. This extra illumination $\Delta I_0(\mathbf{x}_l, \omega_l)$ should diffuse across shadow boundaries as observed in previous work [Pharr and Hanrahan 2000; Jensen et al. 2001].

To compute the correction term of $I_b(\mathbf{x}', \omega_t)$, we emit a large number of photons into the object shell from $\mathbf{x}_l$ in all directions, and then calculate the multiple-scattering irradiance contribution of these photons to the voxel at $\mathbf{x}'$ as

$$\hat{I}_m(\mathbf{x}', \mathbf{x}_l) = \frac{1}{4\pi} \frac{\sum_m \Delta\Phi_p(\mathbf{x}', \omega_p)}{\sigma_s(\mathbf{x}')\Delta V},$$

which is evaluated the same way as the multiple-scattering STF value $I_m(\mathbf{x}', \omega_l)$. The correction to $I_b(\mathbf{x}', \omega_t)$ due to $\mathbf{x}_l$ is thus $\hat{I}_m(\mathbf{x}', \mathbf{x}_l)\Delta I_0(\mathbf{x}_l, \omega_l)$. The total correction to $I_b(\mathbf{x}', \omega_t)$ includes the individual correction term of each $\mathbf{x}_l$ that is in the neighborhood of $\mathbf{x}'_l$ but not in shadow. The neighborhood dimension is set to twice the diffuse mean free path [Jensen et al. 2001]. For the case where $\mathbf{x}'$ is near a shadow boundary but not in shadow, the correction term is computed identically, because the correction is negative when $\Delta I_0(\mathbf{x}_l, \omega_l)$ is negative.

For a given shell surface voxel $\mathbf{x}_l$, we call $\hat{I}_m(\mathbf{x}, \mathbf{x}_l)$ the irradiance correction kernel (ICK) and represent it as a function defined on a 3D array of voxels, called the domain of the ICK. Because the object shell is synthesized from the STF base volume V_b, the ICK can be precomputed with V_b and then applied to the object shell during rendering. Since V_b is non-homogeneous, we should in principle precompute an ICK for each shell surface voxel of V_b. To reduce the number of different ICKs, we cluster the shell surface voxels using the K-means algorithm according to material properties in their ICK domains. A single ICK is then computed and stored for each cluster. The similarity measure for ICK domains is the same as the neighborhood similarity used in shell synthesis.

While higher precision in irradiance correction could be obtained with a larger number of ICKs, for the examples in this paper we simply used one ICK computed by tracing 1000 photons in a homogeneous array. This simplified model typically gives a reasonable irradiance correction, as shown in Fig. 12. The left image is rendered without irradiance correction along the shadow boundary, while the right one is rendered with irradiance correction.

8 Results and Discussion

Our implemented STF modeling and rendering system consists of four components: one for modelling the STF base volume V_b, one for generating a STF sample from V_b using photon tracing, one for synthesizing the STF sample onto a surface, and one for rendering the final STF-based model. The system is fairly easy to use. The main user intervention required is for modelling a small piece of the material sample. Once the STF sample is obtained, the synthesis and rendering steps follow the conventional surface texturing pipeline. In the following we report results for STFs generated from both scan-converted geometry models and CT-scan data and demonstrate various visual effects of non-homogeneous materials with surface mesostructures and volumetric texture variations.

In Fig. 14, we display STF modeling and rendering results for the Stanford bunny under different viewing and lighting conditions. When the light source is behind the bunny, backlighting caused by multiple scattering can be observed. The extinction coefficient κ of the marble is a constant taken from published measurement data [Jensen et al. 2001]. For the albedo map $\alpha(\mathbf{x})$, we generated voxel colors with the sample-view-based synthesis approach [Wei 2001] and converted the color map to the albedo map using the technique proposed in [Jensen and Buhler 2002]. The two input views of the marble sample used for solid texture synthesis are included on the right side of Fig. 14. One view is a texture image of real marble, while the other view is synthesized from the first view using 2D texture synthesis [Wei 2001].

Fig. 15 exhibits a wax torus rendered under different viewing and lighting conditions with an STF obtained from a CT scan of a real volume. Both the mesostructure on the wax surface and the volumetric texture variations within the wax are well captured and rendered. Fig. 10 (a) illustrates a volume rendering result of the original CT data, which is classified into three materials (wax, ball and air). For STF construction, we assign constant extinction coefficients to each material and manually determine the albedo for each

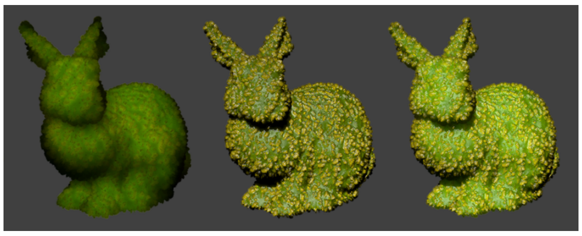

Figure 8: Radiance contributions from different STF components for a bumpy bunny: object shell irradiated by only the inner core contribution (left), the object shell irradiated by only external lighting without the inner core (middle), and the final result (right).

object	object size	V_b size	rendering
jade torus	$304 \times 304 \times 112$	$96 \times 96 \times 10$	84 (38)
bunny	$336 \times 336 \times 272$	$96 \times 96 \times 6$	102 (52)
strawberry	$240 \times 240 \times 272$	$96 \times 96 \times 10$	94 (54)

Table 1: Rendering timings based on our unoptimized implementation. The "rendering" column gives the timings for 720×486 images in seconds. The numbers in parentheses are timings for computing irradiance distributions, including the dipole approximation.

ball. The mesostructure surface is extracted from the CT data as an isosurface [Kaufman 1991]. Finally, we simply mirrored the original sample to form a 2×2 pattern which is used as the STF sample. Note that the balls are completely immersed in the wax and the light we see from the color balls in Fig. 15 completely comes from within the object interior. This is an effect that cannot be generated by a dipole diffusion approximation with surface texture extension [Jensen et al. 2001].

Another set of results is given in Fig. 16, which shows a strawberry rendered under different viewing and lighting conditions. In this example, surface mesostructures include the small dents on the strawberry and the seeds in the dents. Both subsurface scattering and mesostructures are important for the appearance of the strawberry. Subsurface scattering gives the strawberry its distinctive organic look, whereas mesostructures provide the realistic surface details. Its surface mesh shown in Fig. 10 (c) was modeled by an artist. Voxel densities of the STF base volume were scan converted from the meso-geometry model shown in Fig. 10 (d). We hand-tuned a constant κ separately for strawberry seeds and body because we did not find published measurement data for strawberries. As for the albedo map $\alpha(\mathbf{x})$, we generated voxel colors with a procedural approach [Ebert et al. 1994] and then converted the color map to an albedo map using the technique proposed in [Jensen and Buhler 2002].

Fig. 8 exhibits the radiance contributions from the inner core through the object shell, from the object shell as illuminated by only external lighting, and the final rendering result as the sum of the two components.

Two additional STFs are rendered in Fig. 9. Our technique captures both mesostructures and subsurface scattering within non-homogeneous materials of the two samples modelled by an artist. In particular, the mesostructure silhouettes are rendered properly, even for the non height field mesostructure of the bunny.

These examples demonstrate that the STF is an effective technique for modelling a broad range of non-homogeneous materials that have both surface mesostructures and volumetric texture variations. With the STF-based approach, the visual effects that arise from these material and object properties are automatically generated for an object from only a material sample and a given mesh.

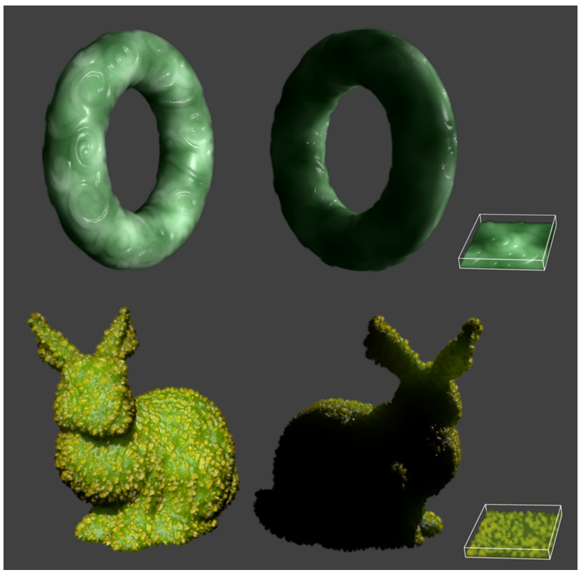

Figure 9: Models rendered with different STFs. Top row: jade torus. Bottom row: bunny with non-height-field bumps.

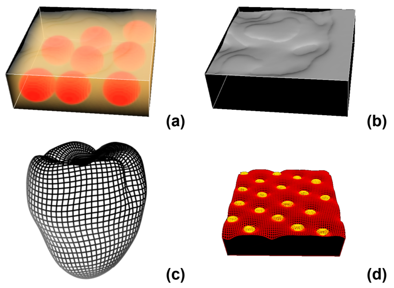

Figure 10: STF modeling. Wax torus of Fig. 15: (a) Direct volume rendering of the original CT data; (b) The outermost isosurface of the wax extracted from the CT data. Strawberry of Fig. 16: (c) Stawberry surface mesh; (d) Meso-geometry model for the base volume of the material sample. The dents are generated as a displacement map, and the seeds (shown in yellow) are manually placed into the dents. The modeling is done using 3D Studio Max.

Rendering times on a 2.4 GHZ Xeon workstation are listed in Table 1 for the bunny, torus and strawberry. The timings indicate that the STF provides an efficient way to capture the visual effects of non-homogeneous materials under dynamic lighting and viewing conditions. Currently, achieving these visual effects would require a full participating media simulation, which is about two orders of magnitude more expensive than the run-time costs of STF rendering (see Fig. 11).

These timings are only preliminary and could be improved significantly. For example, the long irradiance calculation time is due to the very high resolution dipole model we employ, and our diffuse dipole approximation is not accelerated as in [Jensen and Buhler 2002]. The efficiency of STF rendering can be attributed to simple calculations of surface radiance from the precomputed STF data and the fast dipole approximation. The only object areas that require additional computation are those near shadow boundaries, which generally constitute just a small portion of the object. To precompute the STF values by photon mapping, it takes about $30 \sim 50$

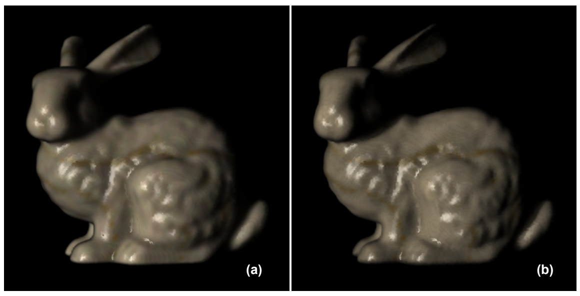

Figure 11: Side-by-side comparison of bunny rendered (a) with photon mapping in 3190 seconds and (b) with the STF in 77 seconds. The results appear similar.

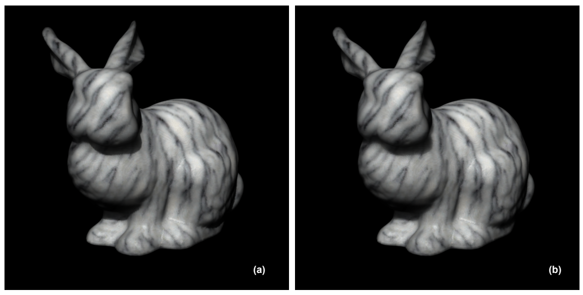

Figure 12: Comparison of the bunny rendered (a) without irradiance correction. (b) with irradiance correction.

minutes for each light direction, which corresponds to 10-20 hours for the examples used in this paper. This procedure, however, can be easily parallelized. On the above-mentioned CPU, synthesis of an STF onto a mesh model takes about half an hour.

Discussion: For an STF-based model, STF rendering generates similar results as photon mapping [Jensen and Christensen 1998] but at much lower costs. For the example presented in Fig. 11, photon mapping requires 3190 seconds and STF rendering takes 77 seconds on a 2.4 GHZ Xeon workstation. The speed increase of STF rendering can be attributed to the fast irradiance computation of STF rendering, which consists of the dipole computation for the inner core and a simple look-up procedure for the material shell.

The homogeneous core approximation is an effective way to handle the inner core of a non-homogeneous object. Fig. 13 exhibits the comparisons of a full photon tracing simulation, the STF-based rendering of varying shell thickness, and a diffuse dipole approximation of the same cube which contains volumetric texture variations. For the dipole approximation shown in (f), material variations within the volume are not modelled and do not appear in the rendering. With an increasing shell thickness, the result of STF rendering becomes closer to that of the full photon mapping solution displayed in (a). A 12-layer STF, shown in (b), well approximates the full photon tracing result for this volume.

Since our technique is intended for optically thick materials in which multiple scattering is predominant, the STF is not suitable for very thin or very translucent objects where the extinction coefficient is small. Also, as described in Section 5.2, the STF should be sufficiently thick according to the scattering properties so that subsurface details are adequately conveyed.

It is important to note that the STF approach addresses both modeling (i.e., authoring) and rendering of objects of non-homogeneous materials. The STF rendering efficiency only applies to STF-based

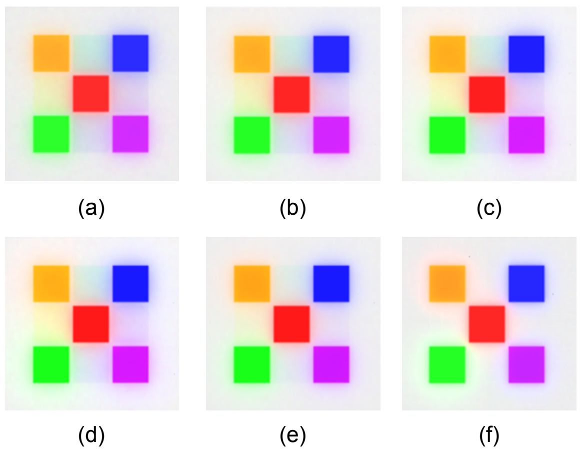

Figure 13: Comparison of different rendering results for a material sample with colored blocks at different depths. (a) full photon mapping, (b) (e) STF with 12, 8, 4, and 2 layers respectively, and (f) dipole approximation. Note that the dipole approximation can only model the color blocks touching the object surface, and not the color blocks that do not touch the top surface. STF rendering appears more and more similar to full photon mapping for increasing STF thicknesses.

models; the STF is not intended to be a rendering technique for arbitrary existing objects of non-homogeneous materials. For rendering such objects, a full participating media simulation is still necessary at present.

9 Conclusion

We have presented a texture function for realistic modeling and efficient rendering of objects of non-homogeneous materials with surface mesostructures and volumetric texture variations. Our results demonstrate that this texture function provides an effective and efficient way to capture the rich visual effects associated with such materials under dynamic lighting and viewing conditions. A number of improvements to our system are possible, such as accelerating the dipole diffusion approximation, more precise processing of non-uniform local illumination, and hardware accelerated rendering. Some other directions for future work include modeling surface curvature in the STF, modulating the material and STF over the mesh to allow for greater modeling flexibility, and improving shadow computation in the spirit of [Dachsbacher and Stamminger 2003].

Acknowledgements

We are grateful to the anonymous SIGGRAPH reviewers, whose comments were very valuable in refining our paper. Many thanks also to the following people for their assistance in this work: Mingdong Xie of D5 Studio for creating the strawberry model, Zhunping Zhang for his help in the early stages of this project and for writing the photon mapping code for STF sampling, Ying Song for her help in processing CT-scanned data, and Prof. Julie Dorsey and Prof. Leonard McMillan for useful discussions.

References

BEERS, A. C., AGRAWALA, M., AND CHADDHA, N. 1996. Rendering from compressed textures. In *Proceedings of SIGGRAPH 1996*, 373–378.

BLINN, J. F. 1978. Simulation of wrinkled surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings) 12*, 3, 286–292.

CHEN, W.-C., BOUGUET, J.-Y., CHU, M. H., AND GRZESZCZUK, R. 2002. Light field mapping: Efficient representation and hardware rendering of surface light fields. *ACM Transactions on Graphics 21*, 3 (July), 447–456.

COOK, R. L. 1984. Shade trees. *Computer Graphics (SIGGRAPH '84 Proceedings) 18*, 3, 223–231.

DACHSBACHER, C., AND STAMMINGER, M. 2003. Translucent shadow maps. In *Proceedings of the 14th Eurographics Workshop on Rendering*, 197–201.

DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., AND KOENDERINK, J. J. 1999. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1 (January), 1–34.

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

DISCHLER, J. M., GHAZANFARPOUR, D., AND FREYDIER, R. 1998. Anisotropic solid texture synthesis using orthogonal 2D views. *Computer Graphics Forum 17*, 3, 87–96.

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

EBERT, D., MUSGRAVE, K., PEACHEY, D., PERLIN, K., AND WORLEY, S. 1994. *Texturing and Modeling: A Procedural Approach*. AP Professional.

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

HAO, X., BABY, T., AND VARSHNEY, A. 2003. Interactive subsurface scattering for translucent meshes. In *ACM Symposium on Interactive 3D Graphics*, 75–82.

HENYEY, L., AND GREENSTEIN, J. 1941. Diffuse radiation in the galaxy. *Astrophysics Journal 93*, 70–83.

JENSEN, H. W., AND BUHLER, J. 2002. A rapid hierarchical rendering technique for translucent materials. In *Proceedings of SIGGRAPH 2002*, 576–581.

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

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

KAUFMAN, A. 1991. *Volume Visualization*. IEEE Computer Society Press.

KOENDERINK, J. J., AND DOORN, A. J. V. 1996. Illuminance texture due to surface mesostructure. *Journal of the Optical Society of America 13*, 3, 452–463.

KOENDERINK, J., AND VAN DOORN, A. 2001. Shading in the case of translucent objects. *Proceedings of SPIE 4299*, 312–320.

LENGYEL, J. E., PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2001. Real-time fur over arbitrary surfaces. In *Symposium on Interactive 3D Graphics*, 227–232.

LEUNG, T., AND MALIK, J. 2001. Representing and recognizing the visual appearance of materials using 3D textons. *International Journal of Computer Vision 43*, 1 (June), 29–44.

LEVOY, M. 1988. Display of surfaces from volume data. *IEEE Computer Graphics & Applications 8*, 3 (May), 29–37.

MALZBENDER, T., GELB, D., AND WOLTERS, H. 2001. Polynomial texture maps. *Proceedings of SIGGRAPH 2001*, 519–528.

MATUSIK, W., PFISTER, H., NGAN, A., BEARDSLEY, P., ZIEGLER, R., AND MCMILLAN, L. 2002. Image-based 3D photography using opacity hulls. *ACM Transactions on Graphics 21*, 3 (July), 427–437.

MERTENS, T., KAUTZ, J., BEKAERT, P., SEIDEL, H.-P., AND REETH, F. V. 2003. Interactive rendering of translucent deformable objects. In *Proceedings of the 14th Eurographics Workshop on Rendering*, 130–140.

NEYRET, F. 1998. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Trans. on Visualization and Computer Graphics 4*, 1, 55–70.

PAYNE, B. A., AND TOGA, A. W. 1992. Distance field manipulation of surface models. *IEEE Computer Graphics & Applications 12*, 1, 65–71.

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

SIEGEL, R., AND HOWELL, J. 1992. *Thermal Radiation Heat Transfer*. Hemisphere Publishing Corp.

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

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics 21*, 3 (July), 665–672.

TURK, G. 2001. Texture synthesis on surfaces. *Proceedings of SIGGRAPH 2001*, 347–354.

WEI, L.-Y. 2001. *Texture Synthesis by Fixed Neighborhood Searching*. Ph.D. Dissertation, Stanford University.

WOOD, D., AZUMA, D., ALDINGER, W., CURLESS, B., DUCHAMP, T., SALESIN, D., AND STUETZLE, W. 2000. Surface light fields for 3D photography. *Proceedings of SIGGRAPH 2000*, 287–296.

WYMAN, D. R., PATTERSON, M. S., AND WILSON, B. C. 1989. Similarity relations for anisotropic scattering in Monte Carlo simulations of deeply penetrating neutral particles. *J. Computational Physics 81*, 137–150.

YING, L., HERTZMANN, A., BIERMANN, H., AND ZORIN, D. 2001. Texture and shape synthesis on surfaces. *Proceedings of 12th Eurographics Workshop on Rendering* (June), 301–312.

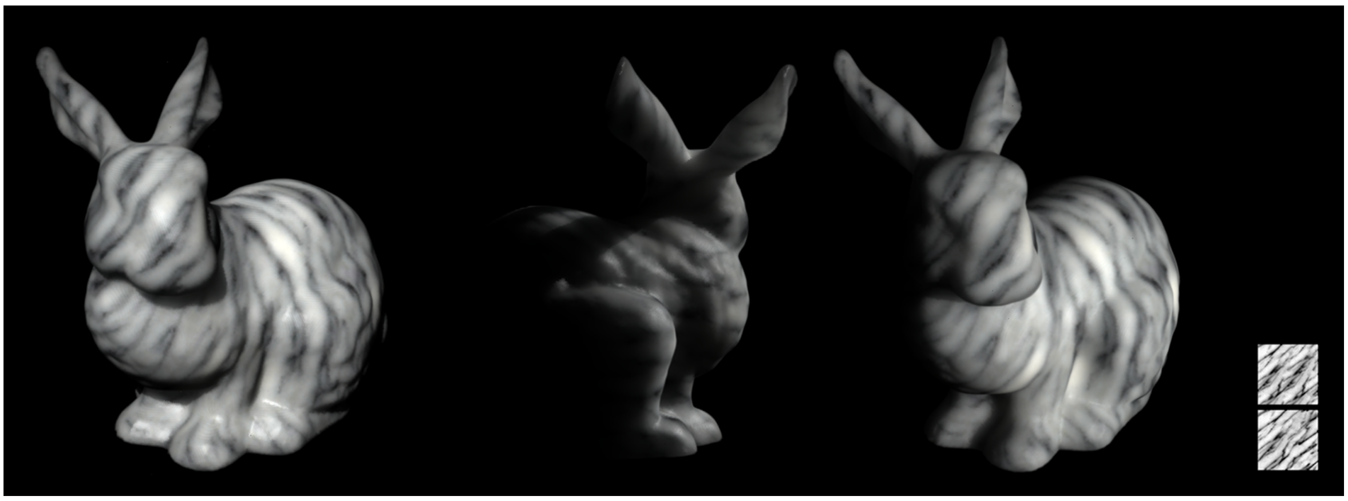

Figure 14: Stanford bunny with marble. The STF used in this example is synthesized from two sample images on the right

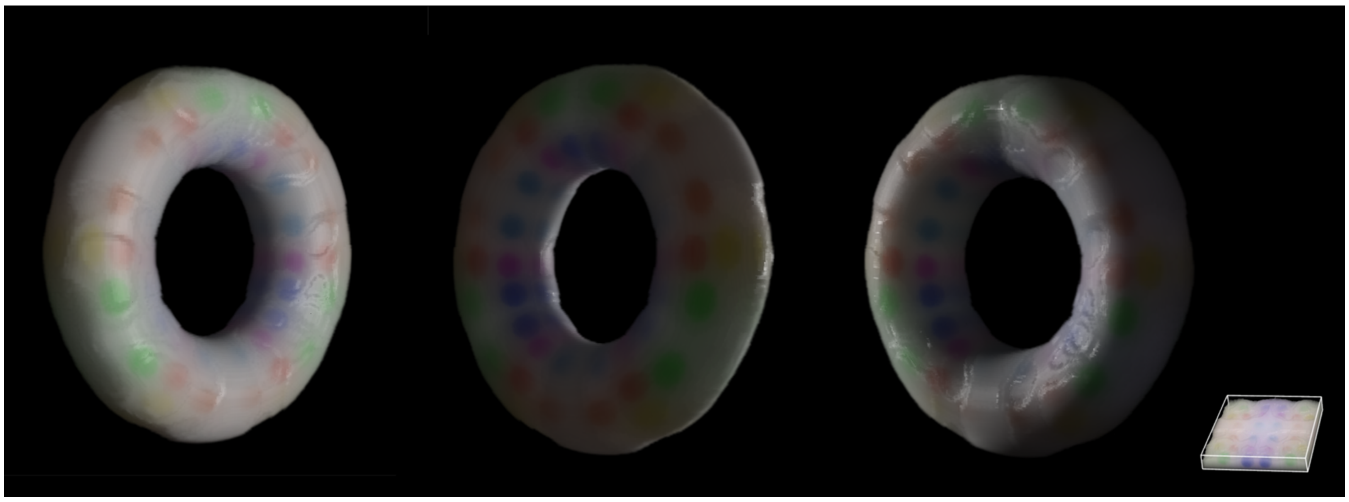

Figure 15: Torus modelled with a wax STF. The STF is obtained from the CT scan of a real volume.

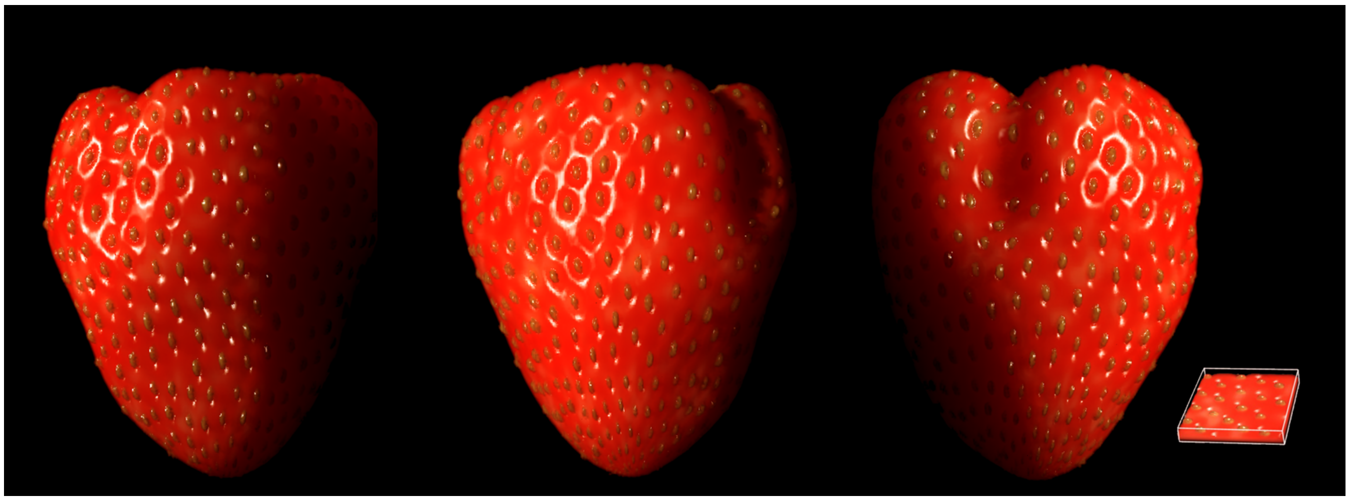

Figure 16: Strawberry rendered by a STF generated from a scan-converted geometry model. Note that both the dents and seeds are part of the surface mesostructure.