

A Model for Volume Lighting and Modeling

Joe Kniss, *Student Member, IEEE Computer Society*,
 Simon Premoze, *Student Member, IEEE Computer Society*,
 Charles Hansen, *Member, IEEE Computer Society*,
 Peter Shirley, *Member, IEEE Computer Society*, and
 Allen McPherson, *Member, IEEE Computer Society*

Abstract—Direct volume rendering is a commonly used technique in visualization applications. Many of these applications require sophisticated shading models to capture subtle lighting effects and characteristics of volumetric data and materials. For many volumes, homogeneous regions pose problems for typical gradient-based surface shading. Many common objects and natural phenomena exhibit visual quality that cannot be captured using simple lighting models or cannot be solved at interactive rates using more sophisticated methods. We present a simple yet effective interactive shading model which captures volumetric light attenuation effects that incorporates volumetric shadows, an approximation to phase functions, an approximation to forward scattering, and chromatic attenuation that provides the subtle appearance of translucency. We also present a technique for volume displacement or perturbation that allows realistic interactive modeling of high frequency detail for both real and synthetic volumetric data.

Index Terms—Volume rendering, shading model, volume modeling, procedural modeling, fur, clouds, volume perturbation.

1 INTRODUCTION

DIRECT volume rendering is widely used in visualization applications. Many of these applications render semi-transparent surfaces lit by an approximation to the Blinn-Phong local surface shading model. This shading model adequately renders such surfaces, but it does not provide sufficient lighting characteristics for translucent materials or materials where scattering dominates the visual appearance. Furthermore, the normal required for the Blinn-Phong shading model is derived from the normalized gradient of the scalar field. While this normal is well-defined for regions in the volume that have high gradient magnitudes, this normal is undefined in homogeneous regions, i.e., where the gradient is the zero vector, as seen in the left side of Fig. 1. The use of the normalized gradient is also troublesome in regions with low gradient magnitudes, where noise can significantly degrade the gradient computation. It has been recently shown that volume rendering techniques can be used to directly visualize multivariate datasets [15]. While a type of derivative measure can be computed for these data sets, it is not suitable for deriving a normal for surface shading. Shadows provide a robust mechanism for shading homogeneous regions in a volume and multivariate field. They also substantially add to the visual perception of volume rendered data, but shadows

are not typically utilized with interactive direct volume rendering because of their high computational expense.

Several studies have shown that the appearance of many common objects is dominated by scattering effects [4], [12]. This is especially true for natural phenomena such as smoke and clouds, but it is also true for wax, skin, and other translucent materials.

While the effects of multiple scattering are important, physically accurate computation of them is not necessarily required for creating meaningful visualizations. In fact, interactive rendering for visualization already often employs such approaches (e.g., ambient light, OpenGL style fog, even the Blinn-Phong surface shading model). Interactivity for visualization is important since it aids in rapidly and accurately setting transfer functions [15], as well as providing important visual cues about spatial relationships in the data. While it is possible to precompute multiple scattering effects, for direct volume rendering, such methods are dependent on the viewpoint, light position, transfer function, and other rendering parameters, which limits interactivity.

The ability to add detail to volumetric data has also been a computational bottleneck. Volume perturbation methods allow such effects at the cost of the time and memory to precompute such details before rendering. Volume perturbation methods have also been employed for modeling natural phenomena such as clouds. Such models, when coupled with a shading model with the visual appearance of scattering, can produce high quality visualizations of clouds and other natural phenomena, as well as introduce visually pleasing details to material boundaries (e.g., direct volume rendered isosurfaces) and statistically appropriate detail for low-resolution volume models.

In a previous paper, we presented a simple yet effective interactive volume shading model that captured effects of volumetric light transport through translucent materials [16].

- J. Kniss and C. Hansen are with the Scientific Computing and Imaging Institute, School of Computing, University of Utah, Salt Lake City, UT 84112. E-mail: {jmk, hansen}@cs.utah.edu.
- S. Premoze and P. Shirley are with the School of Computing, University of Utah, Salt Lake City, UT 84112, E-mail: {premoze, shirley}@cs.utah.edu.
- A. McPherson is with the Advanced Computing Laboratory, Los Alamos National Laboratory, PO Box 1663, Los Alamos, NM 87545. E-mail: mcpherson@lanl.gov.

Manuscript received 16 Nov. 2002; accepted 9 Dec. 2002.

For information on obtaining reprints of this article, please send e-mail to: tcvg@computer.org, and reference IEEECS Log Number SI0013-1102.

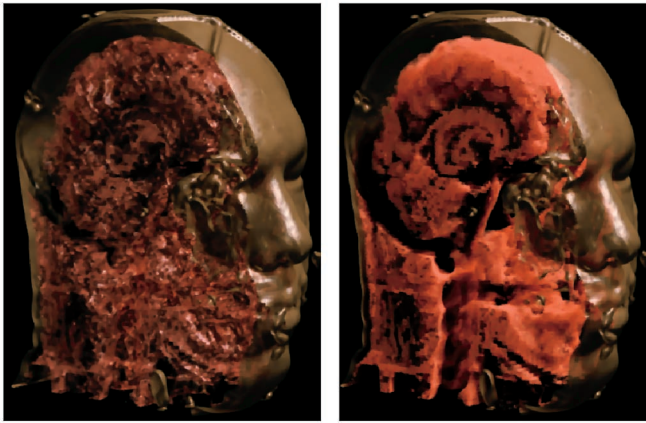


Fig. 1. Surface-based shading versus volume shading model. Surface shading cannot adequately shade homogeneous regions such as the soft tissue in a CT scan, as seen on the left. Our new, more general, volume shading model is needed to render the classified regions, as seen on the right.

In this paper, we more thoroughly describe that shading model and how it relates to the classical volume shading. We also include forward peaked phase function into the model, as well as other applications for volume perturbation. This achieves a method for interactive volumetric light transport that can produce effects such as direct lighting with phase angle influence, volumetric shadows, and a reasonable approximation of multiple scattering. This is shown in Fig. 1. On the left is the standard surface shading of a CT scan of a head. On the right, rendered with the same transfer function, is our improved shading model. Leveraging the new light transport model allows for interactive volume modeling based on volumetric displacement or perturbation. This allows realistic interactive modeling of clouds, height fields, as well as the introduction of details to volumetric data.

In the next section, we introduce the problem of volume shading and light transport and describe related work on volume shading, scattering effects, as well as procedural modeling of clouds and surface detail. We then present the new volumetric shading model which phenomenologically mimics scattering and light attenuation through the volume. Implementation details are discussed and an interactive volumetric perturbation method is introduced. Results on a variety of volume data are also presented to demonstrate the effectiveness of these techniques.

2 BACKGROUND

In this section, we review previous work and give an overview of volume shading equations.

2.1 Previous Work

Volume visualization for scalar fields was described in three papers by Sabella [29], Drebin et al. [7], and Levoy [18]. These methods describe volume shading incorporating diffuse and specular shading by approximating the surface normal with the gradient of the 3D field. The volume rendering described in these seminal papers ignored scattering in favor of the fast approximation achieved by direct lighting. Techniques for implementing

these approaches to volume rendering have been successfully implemented in hardware providing interactive volume rendering of 3D scalar fields [5], [25].

Blinn was one of the first computer graphics researchers to investigate volumetric scattering for computer graphics and visualization applications. He presented a model for the reflection and transmission of light through thin clouds of particles based on probabilistic arguments and single scattering approximation in which Fresnel effects were considered [3]. Kajiya and von Herzen described a model for rendering arbitrary volume densities that included expensive multiple scattering computation. The radiative transport equation [13] cannot be solved analytically except for some simple configurations. Expensive and sophisticated numerical methods must be employed to compute the radiance distribution to a desired accuracy. Finite element methods are commonly used to solve transport equations. Rushmeier [28] and Rushmeier and Torrance [27] presented zonal finite element methods for isotropic scattering in participating media. Max et al. [20] used a one-dimensional scattering equation to compute the light transport in tree canopies by solving a system of differential equations through the application of the Fourier transform. The method becomes expensive for forward peaked phase functions as the hemisphere needs to be more finely discretized. Spherical harmonics were also used by Kajiya and von Herzen [14] to compute anisotropic scattering as well as discrete ordinate methods (Languenou et al. [17]).

Monte Carlo methods are robust and simple techniques for solving light transport equation. Hanrahan and Krueger modeled scattering in layered surfaces with linear transport theory and derived explicit formulas for backscattering and transmission [10]. The model is powerful and robust, but suffers from standard Monte Carlo problems such as slow convergence and noise. Pharr and Hanrahan described a mathematical framework [26] for solving the scattering equation in context of a variety of rendering problems and also described a numerical Monte Carlo sampling method. Jensen and Christensen described a two-pass approach to light transport in participating media [11] using a volumetric photon map. The method is simple, robust, and efficient and it is able to handle arbitrary configurations. Dorsey et al. [6] described a method for full volumetric light transport inside stone structures using a volumetric photon map representation.

Stam and Fiume showed that the often-used diffusion approximation can produce good results for scattering in dense media [31]. Recently, Jensen et al. introduced computationally efficient analytical diffusion approximation to multiple scattering [12], which is especially applicable for homogeneous materials that exhibit considerable subsurface light transport. The model does not appear to be easily extendible to volumes with arbitrary optical properties. Several other specialized approximations have been developed for particular natural phenomena. Nishita et al. [22] presented an approximation to light transport inside clouds and Nishita [21] an overview of light transport and scattering methods for natural environments [21]. These approximations are not generalizable for volume rendering

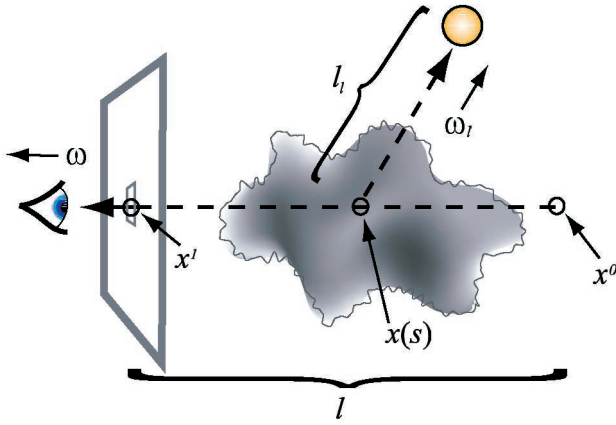


Fig. 2. Geometric setup used in volume shading equations.

applications because of the limiting assumptions made in deriving the approximations.

Max surveyed many optical models for volume rendering applications [19] ranging from very simple to very complex and accurate models that account for all interactions within the volume.

Max [19] and Jensen et al. [12] clearly demonstrate that the effects of multiple scattering and indirect illumination are important for volume rendering applications. However, accurate simulations of full light transport are computationally expensive and do not permit interactivity such as changing the illumination or transfer function. Analytical approximations exist, but they are severely restricted by underlying assumptions, such as homogeneous optical properties and density, simple lighting, or unrealistic boundary conditions. These analytical approximations cannot be used for arbitrary volumes or real scanned data where optical properties such as absorption and scattering coefficients are hard to obtain.

2.2 Volume Shading

The classic volume rendering model originally proposed by Levoy [18] deals with direct lighting only with no shadowing, with the basic geometry shown in Fig. 2. If we parameterize a ray in terms of a distance from the background point \mathbf{x}_0 in direction $\vec{\omega}$, we have:

$$\mathbf{x}(s) = \mathbf{x}_0 + s\vec{\omega} \quad (1)$$

The classic volume rendering model is then written as:

$$L(\mathbf{x}_1, \vec{\omega}) = T(0, l)L(\mathbf{x}_0, \vec{\omega}) + \int_0^l T(s, l)R(\mathbf{x}(s))f_s(\mathbf{x}(s))L_l ds, \quad (2)$$

where R is the surface reflectivity color, f_s is the Blinn-Phong surface shading model evaluated using the normalized gradient of the scalar data field at $\mathbf{x}(s)$, and L_l is the intensity of a point light source. $L(\mathbf{x}_0, \vec{\omega})$ is the background intensity and T the amount the light is attenuated between two points in the volume:

$$T(s, l) = \exp\left(-\int_s^l \tau(s')ds'\right) \quad (3)$$

and $\tau(s')$ is the attenuation coefficient at the sample s' . This volume shading model assumes external illumination from a point light source that arrives at each sample unimpeded by the intervening volume. The only optical properties required for this model are an achromatic attenuation term and the surface reflectivity color, $R(\mathbf{x})$. Naturally, this model is well-suited for rendering surface-like structures in the volume, but performs poorly when attempting to render more translucent materials such as clouds and smoke. Often, the surface lighting terms are dropped and the surface reflectivity color, R , is replaced with the emission term, E :

$$L(\mathbf{x}_1, \vec{\omega}) = T(0, l)L(\mathbf{x}_0, \vec{\omega}) + \int_0^l T(s, l)E(\mathbf{x}(s))ds. \quad (4)$$

This is often referred to as the emission/absorption model. As with the classical volume rendering model, the emission/absorption model only requires two optical properties, α and E . In general, R , from the classical model, and E , from the emission/absorption model, are used interchangeably. This model also ignores inscattering. This means that, although volume elements are emitting light in all directions, we only need to consider how this emitted light is attenuated on its path toward the eye.

Direct light attenuation, or volume shadows, can be added to the classical model as such:

$$L(\mathbf{x}_1, \vec{\omega}) = T(0, l)L(\mathbf{x}_0, \vec{\omega}) + \int_0^l T(s, l)R(\mathbf{x}(s))f_s(\mathbf{x}(s))T_l(s, l_l)L_l ds, \quad (5)$$

where l_l is the distance from the current sample, $\mathbf{x}(s)$, to the light,

$$T_l(t, t') = \exp\left(-\int_0^{t'} \tau(\mathbf{x}(t) + \vec{\omega}_l s)ds\right), \quad (6)$$

and $\vec{\omega}_l$ is the light direction. Just as T represents the attenuation from a sample to the eye, T_l is simply the attenuation from the sample to the light.

Phase function which accounts for directional distribution of light and indirect (inscattered) contributions is still missing in the volume shading model in (5). We will discuss the two missing terms in the following sections. The interested reader is referred to Arvo [1] and Max [19] for in-depth derivations of volume light transport equations and volume shading models and approximations. Important terms used in the paper are found in Table 1.

3 ALGORITHM

Our approximation of the transport equation is designed to provide interactive or near interactive frame rates for volume rendering when the transfer function, light direction, or volume data are not static. Therefore, the light intensity at each sample must be recomputed every frame. Our method for computing light transport is done in image space resolutions, allowing the computational complexity to match the level of detail. Since the computation of light transport is decoupled from the resolution of the volume data, we can also accurately compute lighting for volumes

TABLE 1
Important Terms Used in the Paper

Symbol	Definition
\mathbf{x}	Generic location
s	Distance from the ray's origin
$\mathbf{x}(s)$	Generic location along a ray
R	Surface reflectance
E	Emission in a volume
$T(s, l)$	Attenuation along the ray from $\mathbf{x}(s)$ to $\mathbf{x}(l)$
$\vec{\omega}$	Generic direction
$\vec{\omega}_l$	The light direction
τ	Attenuation coefficient
L_l	Point light source intensity
$L_l(s)$	Light intensity at point $\mathbf{x}(s)$
P	Phase function
l	Generic ray length
l_l	Light ray length

with high frequency displacement effects, which are described in the second half of this section.

3.1 Direct Lighting

Our implementation can be best understood if we first examine the implementation of direct lighting. A brute force implementation of direct lighting, or volumetric shadows, can be accomplished by sending a shadow ray toward the light for each sample along the viewing ray to estimate the amount of extinction caused by the portion of the volume between that sample and the light. This algorithm would have the computational complexity of $O(nm) \equiv O(n^2)$, where n is the total number of samples taken along each viewing ray and m is the number of samples taken along each shadow ray. In general, the algorithm would be far too slow for interactive visualization. It is also very redundant since many of these shadow rays overlap. One possible solution would be to precompute lighting by iteratively sampling the volume from the light's point of view and storing the light intensities at each spatial position in a so-called "shadow volume." While this approach reduces the computational complexity to $O(n + m) \equiv O(n)$, it has a few obvious disadvantages. First, this method can require a significant amount of additional memory for storing the shadow volume. When memory consumption and access times are a limiting factor, one must trade the resolution of the shadow volume and, thus, the resolution of direct lighting computations for reduced memory footprint and improved access times. Another disadvantage of shadow volume techniques is known as "attenuation leakage"; this is caused by the interpolation kernel used when accessing the illumination in the shadow volume. If direct lighting could be computed in lock step with the accumulation of light for the eye, both integrals could be solved iteratively in image space using 2D buffers, one for storing light from the eye's point of view and another for the light source point of view. This can be accomplished using the method of half angle slicing proposed for volume shadow computation in [15], where the slice axis is halfway between the light and view directions or halfway between the light and inverted view directions, depending on the sign of the dot product of the two. The modification of the slicing axis provides the

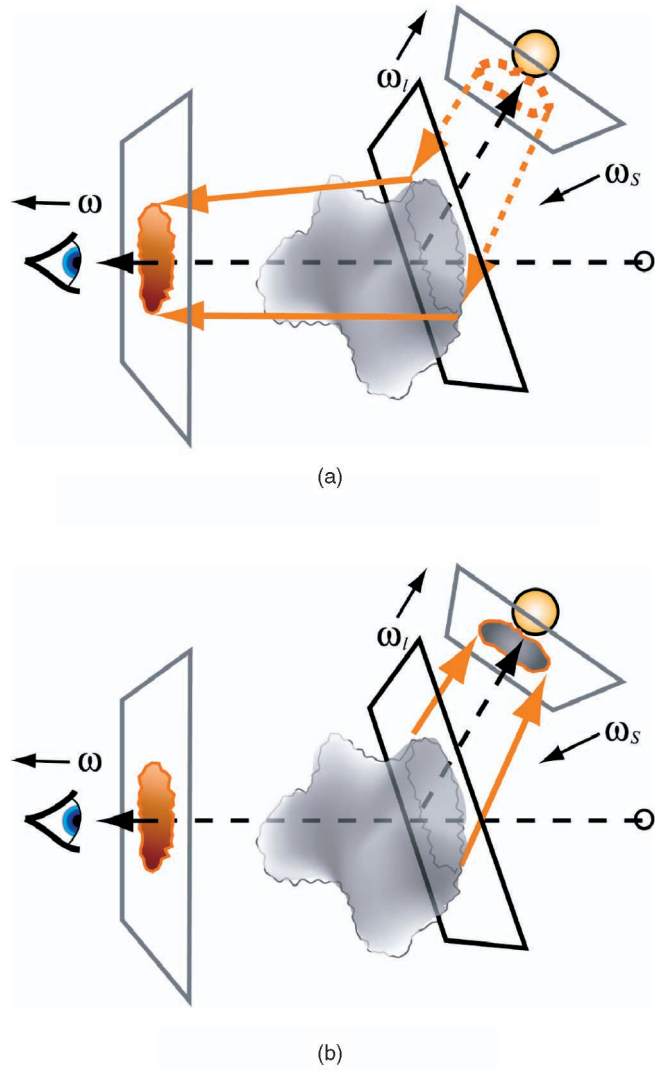


Fig. 3. Two pass shadows using half angle slicing. For each slice, first render into the eye's buffer, sampling the positions that the slice projects to in the light buffer. Next, render the slice into the light buffer, updating the light attenuation for the next slice. $\vec{\omega}_s$ indicates the slice axis. (a) Step 1: Render for eye view, sample light buffer. (b) Step 2: Update light buffer.

ability to render each slice from the point of view of both the observer and the light, thereby achieving the effect of a high resolution shadow map without the requirement of precomputation and storage.

This can be implemented very efficiently on graphics hardware using texture-based volume rendering techniques [33]. The approach requires an additional pass for each slice, which updates the the light intensities for the next slice. The algorithm, by slicing the volume at intervals along some slice axis, $\vec{\omega}_s$, incrementally updates the color and opacity. Each slice is first rendered from the eye's point of view, where the light intensity at each sample on this slice is acquired by sampling the position it would project to in the light's buffer. This light intensity is multiplied by the color of the sample, which is then blended into the eye's buffer. This step is illustrated in Fig. 3a. Next, this slice is rendered into the light buffer, attenuating the light by the opacity at each sample in the slice. This step is illustrated in Fig. 3b.



Fig. 4. An example of a volume rendering with direct lighting.

An example of direct lighting in volume rendering applications can be seen in Fig. 4.

3.2 Phase Functions

The role of the phase function in volume light transport is similar to that of the bidirectional reflectance distribution function (BRDF) in surface-based light transport problems. It describes the distribution after a scattering event for each outgoing direction $\vec{\omega}'$ given an incoming light direction $\vec{\omega}$. While the BRDF is only defined over a hemisphere of directions relative to the normal of the surface, the phase function describes the distribution of light over the entire sphere of directions. Phase function is only dependent on the cosine of the angle between the incoming and outgoing directions $\vec{\omega}$ and $\vec{\omega}'$: $\cos\theta = \vec{\omega} \cdot \vec{\omega}'$. While true phase function is normalized, $\int_{4\pi} P(\vec{\omega}, \vec{\omega}') d\omega' = 1$, we leave our phase functions unnormalized. Fig. 5 shows a plot of a phase function in polar coordinates. The radius r is essentially the weighting for a particular direction. Notice that the phase function is wavelength dependent, indicated by the colored contours. This class of phase functions is referred to as symmetric phase functions because the distribution of scattered energy is rotationally symmetric about the incoming direction. Symmetric phase functions are valid

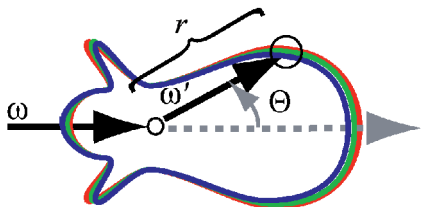


Fig. 5. An example of a symmetric phase function plotted in polar coordinates. The incoming direction $\vec{\omega}$ has fixed direction, while outgoing direction $\vec{\omega}'$ varies over all directions.



Fig. 6. Phase function effects. Images in the left column are rendered without the phase function modulating the direct light contribution, while the images in the right column were rendered with the phase function. Each row has different incoming light direction.

for spherical or randomly oriented particles. For most applications, this class of phase functions is quite adequate.

Symmetrical phase functions can be implemented in conjunction with direct lighting by computing the dot product of the direction from the sample being rendered to the eye with the direction from the light to the sample and then using this scalar value as an index into a one-dimensional lookup table, i.e., this dot product is used as texture coordinate for reading from a 1D texture that stores the fraction of light scattered toward the eye. In our system, we compute these directions for each vertex that defines the corners of the current slice being rendered and apply them as texture coordinates. These coordinates are interpolated over the slice during rasterization. In the per-fragment blending stage, we renormalize these vectors and compute the dot product between them. Since the range of values resulting from the dot product are $[-1..1]$, we first scale and bias the values, so that they are in the range $[0..1]$ and then read from the 1D phase function texture. The result is then multiplied with direct lighting and reflective color. Fig. 6 shows effects of the phase function.

3.3 Indirect Lighting Approximation

Once direct lighting has been implemented in this way, computing the higher order scattering terms becomes a simple extension of this algorithm. As light is propagated from slice to slice, some scattering is allowed. This scattering is forward-only due to the incremental nature of the propagation algorithm. Thus, this is an empirical approximation to the general light transport problem and, therefore, its results must be evaluated empirically.

One major difference between our translucent volume shading model and traditional volume rendering approaches is the additional optical properties required for rendered to simulate higher order scattering. The key to understanding our treatment of optical properties comes from recognizing the difference between absorption and attenuation. The attenuation of light for direct lighting is proportional to extinction, which is the sum of absorption and outscattering.

The traditional volume rendering pipeline only requires two optical properties for each material: attenuation and material color. However, rather than specifying the attenuation term, which is a value in the range zero to infinity, a more intuitive opacity, or alpha, term is used:

$$\alpha = 1 - \exp(-\tau(\mathbf{x})). \quad (7)$$

The material color is the light emitted by the material in the simplified absorption/emission volume rendering model; however, the material color can be thought of as the diffuse reflectance if shadows or surface lighting are included in the model. In addition to these values, our model adds an indirect attenuation term to the transfer function. This term is chromatic, meaning that it describes the indirect attenuation of light for each of the R, G, and B color components. Similar to attenuation, the indirect attenuation can be specified in terms of an indirect alpha:

$$\alpha_i = 1 - \exp(-\tau_i(\mathbf{x})). \quad (8)$$

While this is useful for computing the attenuation, we have found it nonintuitive for user specification. We prefer to specify a *transport color* which is $1 - \alpha_i$ since this is the color the indirect light will become as it is attenuated by the material. Fig. 7 illustrates the difference between the absorption, or indirect alpha, and the transport color. The alpha value can also be treated as a chromatic term (the details of this process can be found in [23]). For simplicity, we treat the alpha as an achromatic value since our aim is to clearly demonstrate indirect attenuation in interactive volume rendering.

Our volume rendering pipeline computes the transport of light through the volume in lock step with the accumulation of light for the eye. Just as we update the direct lighting incrementally, the indirect lighting contributions are computed in the same way. Since we must integrate the incoming light over the cone of incoming directions, we need to sample the light buffer in multiple locations within this cone to accomplish the blurring of light as it propagates.

In the first pass, a slice is rendered from the observer's point of view. In this step, the transfer function is evaluated using a dependent texture read for the reflective color and alpha. In the hardware fragment shading stage, the reflective color is multiplied by the sum of one minus the indirect and direct light attenuation previously computed at that slice position in the **current** light buffer. This color is then blended into the observer buffer using the alpha value from the transfer function.

In the second pass, a slice is rendered into the **next** light buffer from the light's point of view to compute the lighting for the next iteration. Two light buffers are maintained to accommodate the blur operation required for the indirect attenuation. Rather than blend slices using a standard OpenGL blend operation, we explicitly compute the blend in the fragment shading stage. The **current** light buffer is sampled once in the first pass, for the observer, and multiple times in the second pass, for the light, using the *render to texture* OpenGL extension, whereas, the **next** light buffer is rendered to only in the second pass. This relationship changes after the second pass so that the **next**

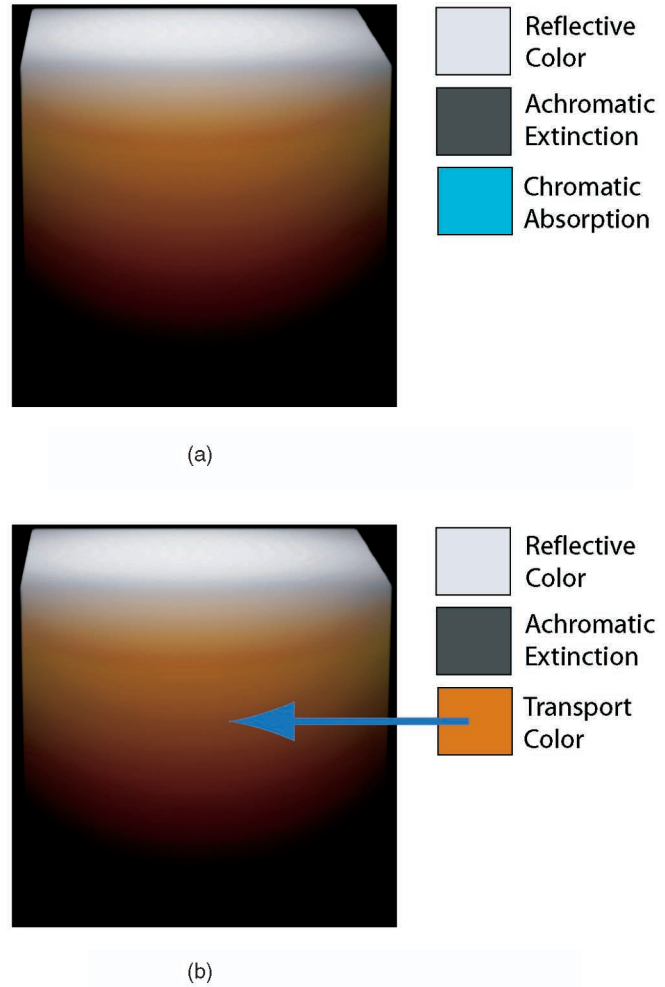


Fig. 7. To attenuate scattered light, we require chromatic indirect alpha, or absorption, term. However, this is very difficult for a user to specify. We prefer specifying the complement of this color, seen in (b), which is the color that light will become as it is attenuated. (a) Optical properties using a chromatic absorption term. (b) Optical properties using the transport color.

buffer becomes the **current** and vice versa. We call this approach *ping pong blending*. In the fragment shading stage, the texture coordinates for the **current** light buffer, in all but one texture unit, are modified per-pixel using a random noise texture, as discussed in the next section. The number of samples used for the computation of the indirect light is limited by the number of texture units. Currently, we use four samples. Randomizing the sample offsets masks some artifacts caused by this coarse sampling. The amount of this offset is restricted to a user-defined blur angle (θ) and the sample distance (d):

$$\text{offset} \leq d \tan\left(\frac{\theta}{2}\right). \quad (9)$$

The **current** light buffer is then read using the new texture coordinates. These values are weighted and summed to compute the blurred inward flux at the sample. The transfer function is evaluated for the incoming slice data to obtain the indirect attenuation (α_i) and direct attenuation (α) values for the current slice. The blurred inward flux is attenuated using α_i and written to the RGB components of

the **next** light buffer. The alpha value from the **current** light buffer with the unmodified texture coordinates is blended with the α value from the transfer function to compute the direct attenuation and stored in the alpha component of the **next** light buffer.

Our empirical volume shading model adds a blurred indirect light contribution at each sample:

$$L(\mathbf{x}_1, \omega) = T(0, l)L(\mathbf{x}_0, \omega) + \int_0^l T(0, s) * C(s) * L_l(s) ds, \quad (10)$$

where $\tau_i(s)$ is the indirect light attenuation term, $C(s)$ is the reflective color at the sample s , $S(s)$ is a surface shading parameter, and L_l is the sum of the direct light and the indirect light contributions. These terms are as follows:

$$C(s) = E(s)((1 - S(s)) + f_s(s)S(s)) \quad (11)$$

$$L_l(s) = L_l * \exp\left(-\int_s^{lt} \tau(x) dx\right) * P(\Theta) + L_l * \exp\left(-\int_s^{lt} \tau_i(x) dx\right) \text{Blur}(\theta), \quad (12)$$

where L_l is the intensity of the light as before, $L_l(s)$ is the light intensity at a location on the ray as it gets attenuated, and $P(\Theta)$ is the phase function. Note that the final model in (10) includes direct and indirect components as well as the phase function that modulates the direct contribution. Spatially varying indirect contribution and phase function were missing in classical volume shading model in (5).

It is interesting to examine the nature of our approximation to the physically-based light transport equation. One way to think of it is as a forward diffusion process. This is different from traditional diffusion approximations [31], [32], [9] because it cannot backpropagate light. A perhaps more intuitive way to think of the approximation is in terms of what light propagation paths are possible. This is shown in Fig. 8. The missing paths involve lateral movements outside the cone or any backscattering. This can give some intuition for what effects our model cannot achieve, such as a reverse bleeding under a barrier. The question of whether the missing paths create a barrier to achieving important visual effects is an empirical one. However, we believe human viewers are not highly sensitive to the details of indirect volume lighting, so there is reason to hope that our approximation is useful.

Because the effect of indirect lighting in dense media is effectively a diffusion of light through the volume, light travels farther in the volume than it would if only direct attenuation is taken into account. Translucency implies blurring of the light as it travels through the medium due to scattering effects. We can approximate this effect by simply blurring the light in some neighborhood and allowing it to attenuate less in the light direction. Fig. 9 shows how the effect of translucency is captured by our model. The upper left image, a wax candle, is an example of a common translucent object. The upper right image is a volume rendering using our model. Notice that the light penetrates much deeper into the material than it does with direct attenuation alone (volumetric shadows), seen in the lower

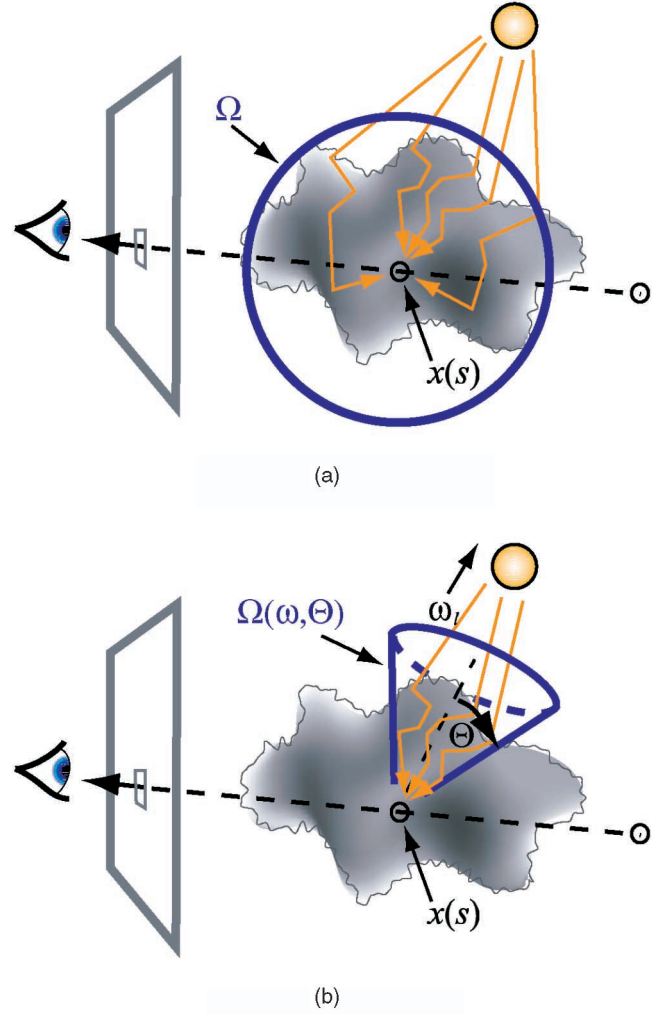


Fig. 8. (a) General light transport scenario, where, at any sample $x(s)$, we must consider incoming light scattered from all directions over the unit sphere Ω . (b) Our approximation, which only considers light scattered in the forward direction within the cone of directions, the light direction ω_l with apex angle Θ .

right image. Also notice the pronounced hue shift from white to orange to black due to an indirect attenuation term that attenuates blue slightly more than red or green. The lower left image shows the effect of changing just the reflective color to a pale blue.

Surface shading can also be added for use with scalar data sets. For this, we recommend the use of a surface shading parameter. This is a scalar value between one and zero that describes the degree to which a sample should be surface shaded. It is used to interpolate between surface shading and no surface shading. This value can be added to the transfer function, allowing the user to specify whether or not a classified material should be surface shaded. It can also be set automatically using the gradient magnitude at the sample, as in [15]. Here, we assume that classified regions will be surface-like if the gradient magnitude is high and therefore should be shaded as such. In contrast, homogeneous regions, which have low gradient magnitudes, should only be shaded using light attenuation.

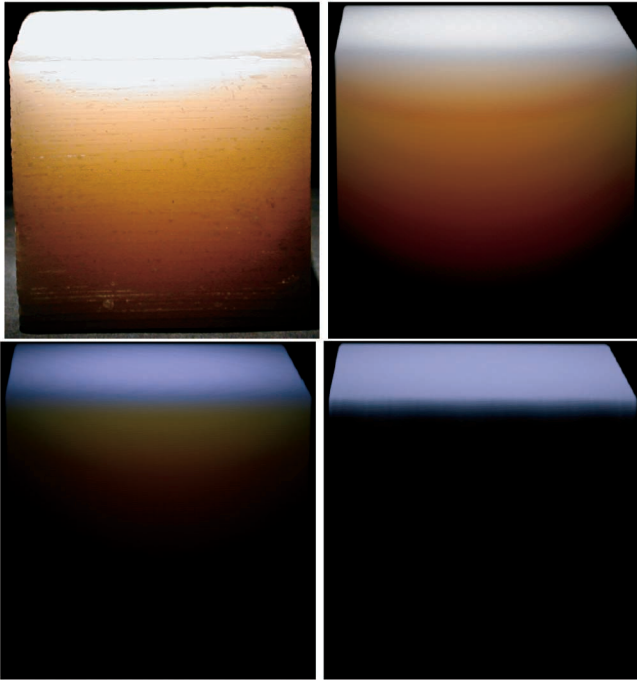


Fig. 9. Translucent volume shading. The upper left image is a photograph of wax block illuminated from above with a focused flashlight. The upper right image is a volume rendering with a white reflective color and a desaturated orange transport color ($1 - \text{indirect attenuation}$). The lower left image has a bright blue reflective color and the same transport color as the upper right image. The lower right image shows the effect of light transport that only takes into account direct attenuation.

3.4 Volume Perturbation

One drawback of volume-based graphics is that high frequency details cannot be represented in small volumes. These high frequency details are essential for capturing the characteristics of many volumetric objects such as clouds, smoke, trees, hair, and fur. Procedural noise simulation is a very powerful tool to use with small volumes to produce visually compelling simulations of these types of volumetric objects. Our approach is similar to Ebert et al.'s approach for modeling clouds [8]: Use a coarse technique for modeling the macrostructure and use procedural noise-based simulations for the microstructure. We have adapted this approach to interactive volume rendering through two volume perturbation approaches which are efficient on modern graphics hardware. The first approach is used to perturb optical properties in the shading stage, while the second approach is used to perturb the volume itself.

Both volume perturbation approaches employ a small 3D perturbation volume, 32^3 . Each texel is initialized with four random 8-bit numbers, stored as RGBA components, and blurred slightly to hide the artifacts caused by trilinear interpolation. Texel access is then set to repeat. An additional pass is required for both approaches due to limitations imposed on the number of textures which can be simultaneously applied to a polygon and the number of sequential dependent texture reads permitted. The additional pass occurs before the steps outlined in the previous section. Multiple copies of the noise texture are applied to each slice at different scales. They are then weighted and

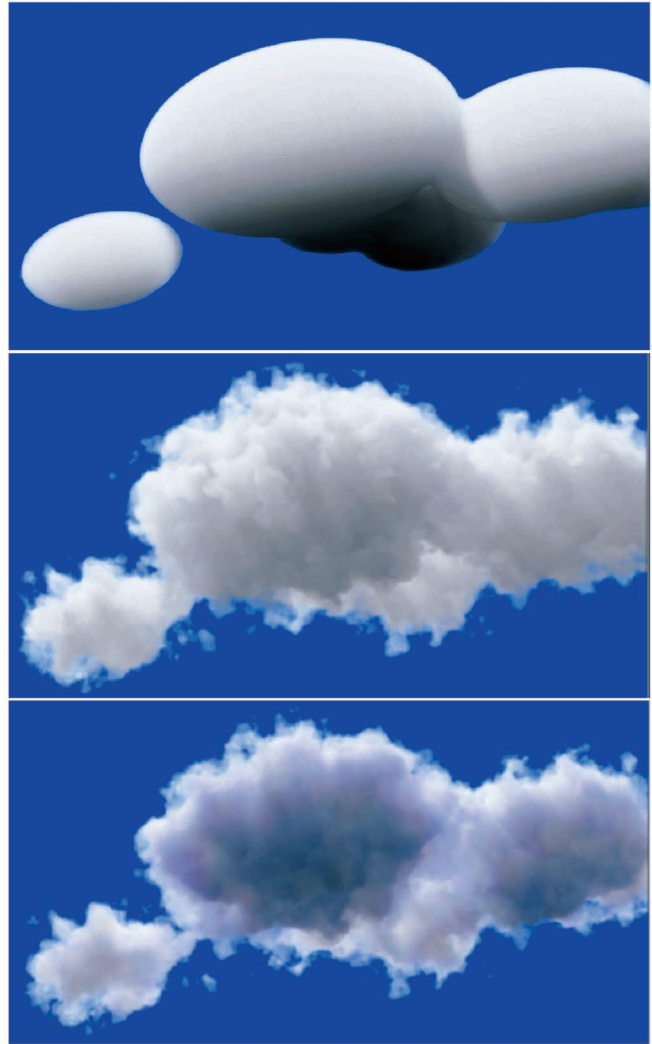


Fig. 10. Procedural clouds. The image on the top shows the underlying data, 64^3 . The center image shows the perturbed volume. The bottom image shows the perturbed volume lit from behind with low frequency noise added to the indirect attenuation to achieve subtle iridescence effects.

summed per pixel. To animate the perturbation, we add a different offset to each noise texture's coordinates and update it each frame.

Our first approach is similar to Ebert et al.'s lattice-based noise approach [8]. It uses the four per-pixel noise components to modify the optical properties of the volume after the the transfer function has been evaluated. This approach makes the materials appear to have inhomogeneities. We allow the user to select which optical properties are modified. This technique is used to get the subtle iridescence effects seen in Fig. 10 (bottom).

Our second approach is closely related to Ebert et al.'s vector-based noise simulation technique [8]. It uses the noise to modify the location of the data access for the volume. In this case, three components of the noise texture form a vector, which is added to the texture coordinates for the volume data per pixel. The data is then read using a dependent texture read. The perturbed data is rendered to a pixel buffer that is used instead of the original volume data. Fig. 11 illustrates this process. (a) shows the original texture

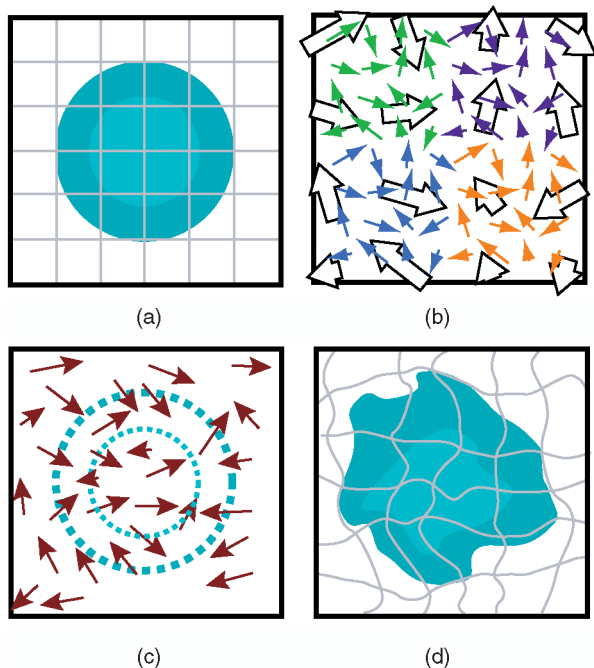


Fig. 11. Texture coordinate perturbation in 2D. (a) shows a square polygon mapped with an unperturbed texture. (b) shows a low resolution vector noise texture applied to the polygon multiple times at different scales to achieve low frequency, high amplitude, offsets (large arrows), and high frequency, low amplitude offsets (small colored arrows). These offset vectors are weighted and summed to offset the original texture coordinates, as seen in (c). The texture is then read using the modified texture coordinates, producing the image seen in (d).

data. (b) shows how the perturbation texture is applied to the polygon twice, once to achieve low frequency with high amplitude perturbations (large arrows) and again to achieve high frequency with low amplitude perturbations (small arrows). Notice that the high frequency content is created by allowing the texture to repeat. Fig. 11c shows the resulting texture coordinate perturbation field when the multiple displacements are weighted and summed. (d) shows the image generated when the texture is read using the perturbed texture coordinates. Fig. 10 shows how a coarse volume model can be combined with our volume perturbation technique to produce an extremely detailed interactively rendered cloud. The original 64^3 voxel dataset is generated from a simple combination of volumetric blended implicit ellipses and defines the cloud macro-structure [8]. The final rendered image in Fig. 10c, produced with our volume perturbation technique, shows detail that would be equivalent to unperturbed voxel dataset of at least one hundred times the resolution. Fig. 12 demonstrates this technique on another example. By perturbing the volume with a high frequency noise, we can obtain a fur-like surface on the Teddy bear.

Another application of spatial perturbation is height field rendering. In this case, the data, a thin volumetric plane, is purely implicit. The perturbation vector field is stored as a single scalar value, the length of the perturbation, in a 2D texture. For height field rendering, the perturbation only occurs in one direction for all points in the volume. The algorithm for generating slices and computing lighting is nearly identical to the algorithm presented earlier in this

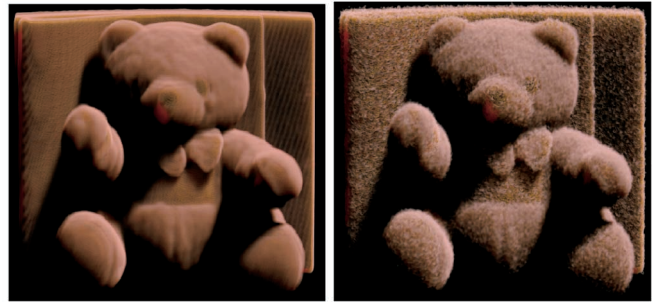


Fig. 12. Procedural fur. Left: Original Teddy bear CT scan. Right: Teddy bear with fur created using high frequency texture coordinate perturbation.

paper. The difference is that the data comes from a 2D texture rather than a 3D volume. In the per-fragment blending stage, the s and t texture coordinates of a fragment, which may have been perturbed using the texture coordinate perturbation approach described above, are used to read from a 2D texture containing the scalar height value and either RGB color values or scalar data values, which can, in turn, be used to acquire optical properties from the transfer function textures. The height value is then subtracted from the r texture coordinate of the fragment to determine whether or not this modified position is within the implicitly defined thin volume. This thin volume is defined using two scalar values that define its top and bottom positions along the r axis of the volume. It may seem natural to simply set the opacity of a fragment to 1 if it is within the thin volume and 0 otherwise; however, this can produce results with very poor visual quality. Rather, it is important to taper the opacity off smoothly at the edges of the implicit plane. We do this by setting an additional scalar value that determines the distance of linear ramp from an opacity of one to zero opacity based on the distance the of fragments $r - h$ position from the top or bottom edge of the implicit thin volume. Fig. 13 shows two examples of volume rendered height fields. Both volumes were generated from 2D height textures with dimensions $1,024 \times 1,024$. One important optimization for volume rendering height fields is mip mapping, which allows automatic level of detail control and a significant performance improvement when volume rendering high resolution height maps.

4 RESULTS AND DISCUSSION

We have implemented our volume shading model on both the NVIDIA GeForce 3 and the ATI Radeon 8500/9500. By taking advantage of the OpenGL *render to texture* extension, which allows us to avoid many time consuming copy to texture operations, we have attained frame rates which are only 50 to 60 percent slower than volume rendering with no shading at all. The frame rates for volume shading are comparable to volume rendering with surface shading (e.g., Blinn-Phong shading). Even though surface shading does not require multiple passes on modern graphics hardware, the cost of the additional 3D texture reads for normals induces a considerable performance penalty compared to the 2D texture reads required for our two pass approach. Rendering times for a sample data set are shown in Fig. 14. The latest

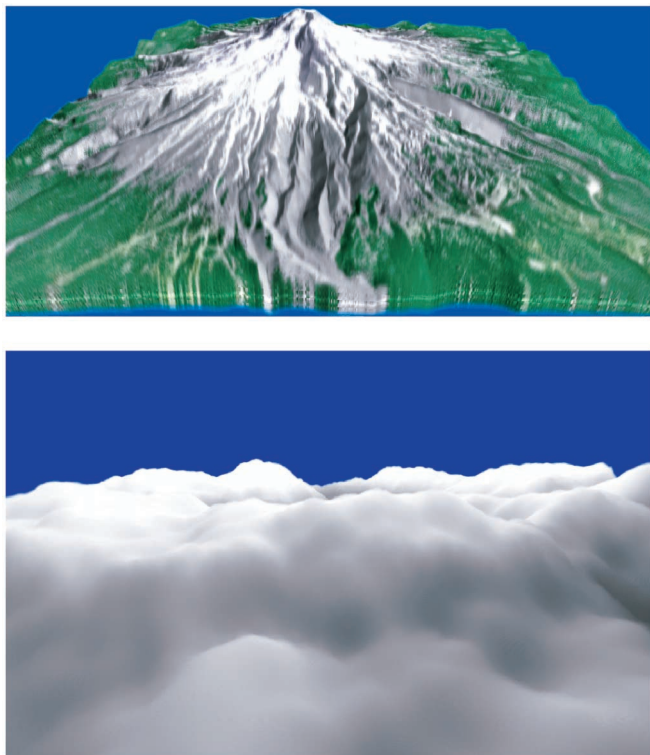


Fig. 13. Height field volume rendering. The top image shows a height field rendering from geographical height data of Mt. Hood. Bottom shows a rendering of a cumulus cloud field generated using a height field.

generations of graphics hardware, such as the ATI Radeon 9500, have very flexible fragment shading capabilities, which allow us to implement the entire shading and perturbation pipeline in a single pass. One issue with using the dual buffers required by our method is the problem of switching OpenGL contexts. We can avoid this by employing a larger render target and using the viewport command to render to subregions within this, as shown by Fig. 15.

While our volume shading model is not as accurate as other more time consuming software approaches, the fact that it is interactive makes it an attractive alternative. Accurate physically-based simulations of light transport require material optical properties to be specified in terms of scattering and absorption coefficients. Unfortunately, these values are difficult to acquire. There does not yet exist a comprehensive database of common material optical

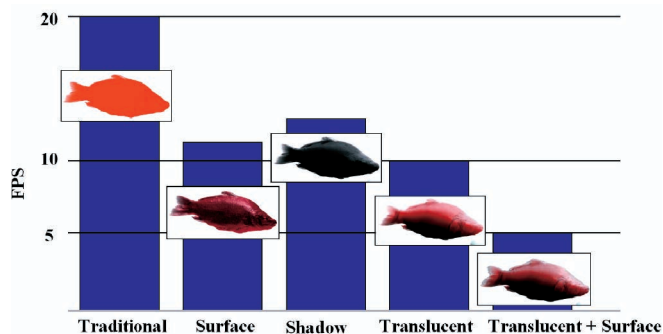


Fig. 14. Rendering times for the shading models discussed in this paper for the CT carp data set.

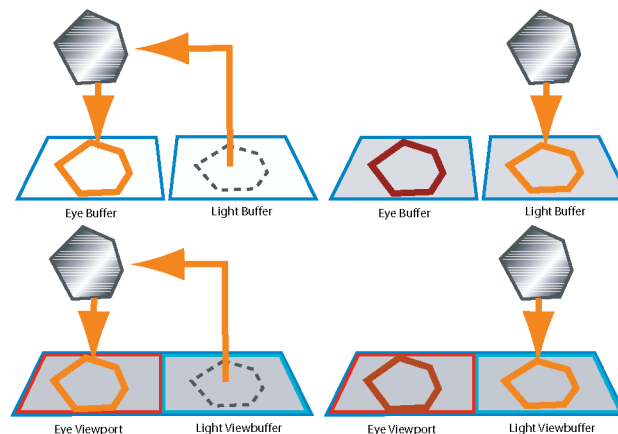


Fig. 15. Avoiding OpenGL context switching. The top image shows the standard approach for multipass rendering: Render into the eye buffer using the results from a previous pass (left) in the light buffer, then update the light buffer. Unfortunately, switching between buffers can be a very expensive operations. We avoid this buffer switch by using a single buffer that is broken into pieces using viewports (seen in the bottom).

properties. Interactivity combined with a higher level description of optical properties (e.g., diffuse reflectivity, indirect attenuation, and alpha) allow the user the freedom to explore and create visualizations that achieve a desired effect. Fig. 16 (top) demonstrates the familiar appearance of

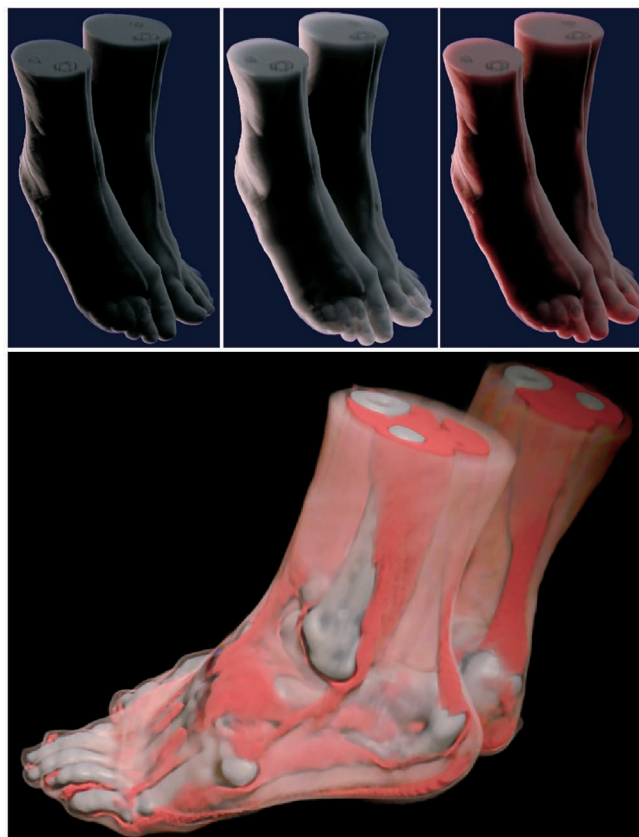


Fig. 16. The feet of the Visible Female CT. The top left image shows a rendering with direct lighting only, the top center image shows a rendering with achromatic indirect lighting, and the top right image shows a rendering with chromatic indirect lighting.

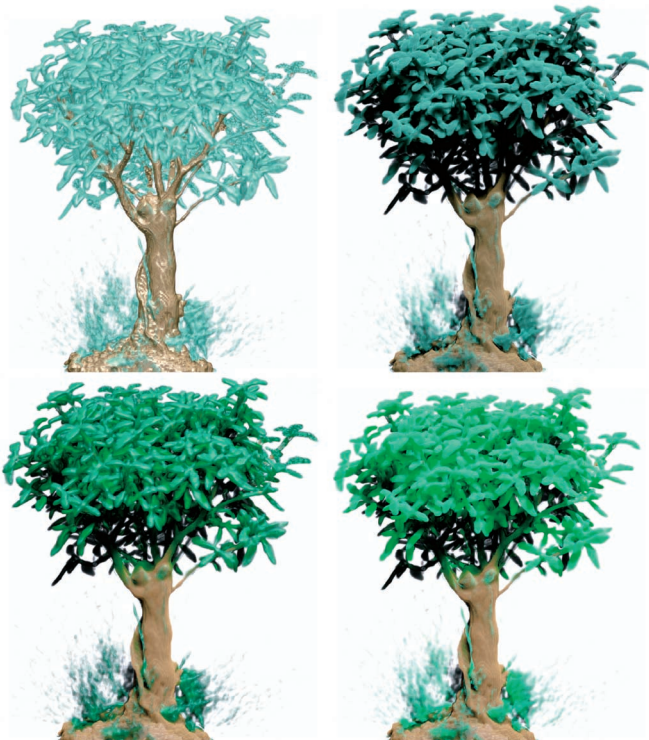


Fig. 17. A comparison of shading techniques. Upper left: surface shading only, upper right: direct lighting only (shadows), lower right: direct and indirect lighting, lower left: direct and indirect lighting with surface shading only on leaves.

skin and tissue. The optical properties for these illustrations were specified quickly (in less than 5 minutes) without using measured optical properties. Even if a user has access to a large collection of optical properties, it may not be clear how to customize them for a specific look. Fig. 16 (bottom) demonstrates the effectiveness of our lighting model for scientific visualization.

Our approach is advantageous over previous hardware volume shadow approaches [2], [24], [30] in several ways. First, since this method computes and stores light transport in image space resolution rather than in an additional 3D texture, we avoid an artifact known as attenuation leakage. This can be observed as materials which appear to shadow themselves and blurry shadow boundaries caused by the trilinear interpolation of lighting stored on a coarse grid. Second, even if attenuation leakage is accounted for, volume shading models which only compute direct attenuation (shadows) will produce images which are much darker than intended. These approaches often compensate for this by adding a considerable amount of ambient light to the scene, which may not be desirable. The addition of indirect lighting allows the user to have much more control over the image quality. All of the images in this paper were generated without ambient lighting. Although the new model does not have a specular component, it is possible to include surface shading specular highlights where appropriate. Such instances include regions where the gradient magnitude is high and there is a zero crossing of the second derivative [15]. Fig. 17 compares different lighting models. All of the renderings use the same color map and alpha values. The image on the upper left is a typical volume



Fig. 18. Example material shaders. Rows: gray, red, green, and blue transport colors, respectively. Columns: white, red, green, and blue reflective colors, respectively. Bottom row: Different noise frequencies: low, low plus medium, low plus med plus high, and just high frequencies, respectively.



Fig. 19. The "Chia Skull." A comparison of shading techniques on the Visible Male skull using texture coordinate perturbation. Upper left: no shading. Upper right: shadows. Lower right: shadows with a lower opacity skull. Lower left: indirect and direct lighting.

rendering with surface shading using the Blinn-Phong shading model. The image on the upper right shows the same volume with only direct lighting, providing volumetric shadows. The image on the lower right uses both direct and indirect lighting. Notice how indirect lighting brightens up the image. The image on the lower left uses direct and indirect lighting combined with surface shading where surface shading is only applied to the leaves where there is a distinct material boundary. Fig. 18 shows several examples of translucent shading. The columns vary the transport color, or the indirect attenuation color, and the rows vary the reflective color, or simply the materials color. This illustration demonstrates only a small subset of the shading effects possible with our model.

Procedural volumetric perturbation provides a valuable mechanism for volume modeling effects such as the clouds seen in Fig. 10 and for adding high frequency details which may be lost in the model acquisition process, such as the fur of the Teddy bear in Fig. 12. Its value in producing realistic effects, however, is largely dependent on the shading. As you can imagine, the clouds in Fig. 10 would look like nothing more than deformed blobs with a surface-based shading approach. By combining a realistic shading model with the perturbation technique, we can achieve a wide range of interesting visual effects. The importance of having a flexible and expressive shading model for rendering with procedural effects is demonstrated in Fig. 19. This example attempts to create a mossy or leafy look on the Visible Male's skull. The upper left image shows the skull with texture coordinate perturbation and no shading. To shade such a perturbed volume with surface shading, one would need to recompute the gradients based upon the perturbed grid. The upper right image adds shadows. While the texture is readily apparent in this image, the lighting is far too dark and harsh for a leafy appearance. The lower right image shows the skull rendered with shadows using a lower alpha value. While the appearance is somewhat brighter, it still lacks the luminous quality of leaves. By adding indirect lighting, as seen in the lower left image, we not only achieve the desired brightness, but we also see the characteristic hue shift of translucent leaves or moss.

5 FUTURE WORK

The lighting model presented in this paper was designed to handle volume rendering with little or no restrictions on external lighting, transfer function, or volume geometry setup. However, if some assumptions can be made, the model can be modified to gain better performance for special purpose situations. We will be exploring extensions of this model that are tailored for specific phenomena or effects, such as clouds, smoke, and skin.

We are also interested in developing more accurate simulations of volumetric light transport that can leverage the expanding performance and features of modern graphics hardware. Such models would be useful for high quality offline rendering as well as the qualitative and quantitative assessment of our current lighting model, thereby guiding future improvements. As the features of programmable graphics hardware become more flexible and general, we look forward to enhancing our model with

effects such as refraction, caustics, back scattering, and global illumination.

Our work with volume perturbation has given us valuable insight into the process of volume modeling. We have been experimenting with approaches for real-time volume modeling which do not require any underlying data. We will be developing implicit volume representations and efficient simulations for interactive applications. We are also exploring the use of volume perturbation in the context of uncertainty visualization, where regions of a volume are deformed based on uncertainty or accuracy information.

ACKNOWLEDGMENTS

The authors wish to acknowledge their coauthor David Ebert who removed his name from the author list due to his editorship of this IEEE transactions. His contribution to the IEEE Visualization paper made this journal paper possible. This material is based upon work supported by the US National Science Foundation under Grants: NSF ACI-0081581, NSF ACI-0121288, NSF IIS-0098443, NSF ACI-9978032, NSF MRI-9977218, NSF ACR-9978099, and the US Department of Energy VIEWS program.

REFERENCES

- [1] J. Arvo, "Transfer Equations in Global Illumination," *Global Illumination, SIGGRAPH '93 Course Notes*, Aug. 1993.
- [2] U. Behrens and R. Ratering, "Adding Shadows to a Texture-Based Volume Renderer," *Proc. 1998 Volume Visualization Symp.*, pp. 39-46, 1998.
- [3] J.F. Blinn, "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces," *Proc. SIGGRAPH '82*, pp. 21-29, 1982.
- [4] C.F. Bohren, "Multiple Scattering of Light and Some of Its Observable Consequences," *Am. J. Physics*, vol. 55, no. 6, pp. 524-533, June 1987.
- [5] B. Cabral, N. Cam, and J. Foran, "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware," *Proc. 1994 Symp. Volume Visualization*, A. Kaufman and W. Krueger, eds., pp. 91-98, Oct. 1994.
- [6] J. Dorsey, A. Edelman, H.W. Jensen, J. Legakis, and H. Pedersen, "Modeling and Rendering of Weathered Stone," *Proc. SIGGRAPH '99*, pp. 225-234, Aug. 1999.
- [7] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graphics (SIGGRAPH '88 Proc.)*, J. Dill, ed., vol. 22, pp. 65-74, Aug. 1988.
- [8] D. Ebert, F.K. Musgrave, D. Peachey, K. Perlin, and S. Worley, *Texturing and Modeling: A Procedural Approach*. Academic Press, July 1998.
- [9] T.J. Farrell, M.S. Patterson, and B.C. Wilson, "A Diffusion Theory Model of Spatially Resolved, Steady-State Diffuse Reflectance for the Non-Invasive Determination of Tissue Optical Properties in Vivo," *Medical Physics*, vol. 19, pp. 879-888, 1992.
- [10] P. Hanrahan and W. Krueger, "Reflection from Layered Surfaces Due to Subsurface Scattering," *Computer Graphics (SIGGRAPH '93 Proc.)*, J.T. Kajiya, ed., vol. 27, pp. 165-174, Aug. 1993.
- [11] H.W. Jensen and P.H. Christensen, "Efficient Simulation of Light Transport in Scenes with Participating Media Using Photon Maps," *Proc. SIGGRAPH '98, Computer Graphics Proc., Ann. Conf. Series*, pp. 311-320, 1998.
- [12] H.W. Jensen, S.R. Marschner, M. Levoy, and P. Hanrahan, "A Practical Model for Subsurface Light Transport," *Proc. SIGGRAPH 2001, Computer Graphics Proc., Ann. Conf. Series*, pp. 511-518, Aug. 2001.
- [13] J.T. Kajiya, "The Rendering Equation," *Computer Graphics (SIGGRAPH '86 Proc.)*, D.C. Evans and R.J. Athay, eds., vol. 20, pp. 143-150, Aug. 1986.

- [14] J.T. Kajiya and B.P. Von Herzen, "Ray Tracing Volume Densities," *Computer Graphics (SIGGRAPH '84 Proc.)*, H. Christiansen, ed., vol. 18, pp. 165-174, July 1984.
- [15] J. Kniss, G. Kindlmann, and C. Hansen, "Multidimensional Transfer Functions for Interactive Volume Rendering," *IEEE Trans. Visualization and Computer Graphics*, vol. 8, no. 3 pp. 270-285, July-Sept. 2002.
- [16] J. Kniss, S. Premoze, C. Hansen, and D. Ebert, "Interactive Volume Light Transport and Procedural Modeling," *Proc. IEEE Visualization 2002*, pp. 109-116, 2002.
- [17] E. Languenou, K. Bouatouch, and M. Chelle, "Global Illumination in Presence of Participating Media with General Properties," *Proc. Fifth Eurographics Workshop Rendering*, pp. 69-85, June 1994.
- [18] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications*, vol. 8, no. 5, pp. 29-37, 1988.
- [19] N. Max, "Optical Models for Direct Volume Rendering," *IEEE Trans. Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99-108, June 1995.
- [20] N. Max, C. Mobley, B. Keating, and E. Wu, "Plane-Parallel Radiance Transport for Global Illumination in Vegetation," *Proc. Eurographics Rendering Workshop 1997*, J. Dorsey and P. Slusallek, eds., pp. 239-250, June 1997.
- [21] T. Nishita, "Light Scattering Models for the Realistic Rendering," *Proc. Eurographics*, pp. 1-10, 1998.
- [22] T. Nishita, E. Nakamae, and Y. Dobashi, "Display of Clouds and Snow Taking into Account Multiple Anisotropic Scattering and Sky Light," *SIGGRAPH '96 Conf. Proc., Ann. Conf. Series*, H. Rushmeier, ed., pp. 379-386, Aug. 1996.
- [23] H.J. Noordmans, H.T. van der Voort, and A.W. Smeulders, "Spectral Volume Rendering," *IEEE Trans. Visualization and Computer Graphics*, vol. 6, no. 3, July-Sept. 2000.
- [24] M. Nulkar and K. Mueller, "Splatting with Shadows," *Proc. Volume Graphics 2001*, pp. 35-49, 2001.
- [25] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, "The VolumePro Real-Time Ray-Casting System," *ACM Computer Graphics (SIGGRAPH '99 Proc.)*, pp. 251-260, Aug. 1999.
- [26] M. Pharr and P.M. Hanrahan, "Monte Carlo Evaluation of Non-Linear Scattering Equations for Subsurface Reflection," *Proc. SIGGRAPH 2000, Computer Graphics Proc., Ann. Conf. Series*, pp. 75-84, July 2000.
- [27] H.E. Rushmeier, "Realistic Image Synthesis for Scenes with Radiatively Participating Media," PhD thesis, Cornell Univ., 1988.
- [28] H.E. Rushmeier and K.E. Torrance, "The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium," *Computer Graphics (SIGGRAPH '87 Proc.)*, M.C. Stone, ed., vol. 21, pp. 293-302, July 1987.
- [29] P. Sabella, "A Rendering Algorithm for Visualizing 3D Scalar Fields," *Computer Graphics (SIGGRAPH '88 Proc.)*, J. Dill, ed., vol. 22, pp. 51-58, Aug. 1988.
- [30] J. Stam, "Stable Fluids," *Proc. Siggraph '99*, pp. 121-128, 1999.
- [31] J. Stam and E. Fiume, "Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes," *SIGGRAPH '95 Conf. Proc., Ann. Conf. Series*, R. Cook, ed., pp. 129-136, Aug. 1995.
- [32] L.V. Wang, "Rapid Modelling of Diffuse Reflectance of Light in Turbid Slabs," *J. Optical Soc. Am. A*, vol. 15, no. 4, pp. 936-944, 1998.
- [33] O. Wilson, A.V. Gelder, and J. Wilhelms, "Direct Volume Rendering via 3D Textures," Technical Report UCSC-CRL-94-19, Univ. of California at Santa Cruz, June 1994.



member of the IEEE Computer Society.



worked on computer simulation and visualization of liquid crystal phase transitions and dynamics of liquid crystals. He is a student member of the IEEE Computer Society.



National Laboratory, where he formed and directed the visualization efforts in the ACL. He was a Bourse de Chateaubriand PostDoc Fellow at INRIA, Rocquencourt France, in 1987 and 1988. His research interests include large-scale scientific visualization and computer graphics. He is a member of the IEEE Computer Society.



include computer graphics, statistical computing, visualization, and immersive environments. He is a member of the IEEE Computer Society.



Joe Kniss received the BS degree in 1999 from Idaho State University and the MS degree in computer science from the University of Utah in 2002. He is currently pursuing a PhD degree in computer science at the University of Utah, where he is a member of the Scientific Computing and Imaging Institute. His research interests include computer graphics, light transport in participating media, human computer interaction, immersive environments. He is a student

Simon Premoze is a PhD student in the School of Computing at the University of Utah working under the supervision of Professor Peter Shirley. He obtained the BS degree in computer science from the University of Colorado at Boulder. His research interests include global illumination and rendering algorithms, modeling natural phenomena, reflectance models, and mountain cartography. He has done research on modeling and rendering natural phenomena. Previously, he

Charles Hansen received the BS degree in computer science from Memphis State University in 1981 and the PhD degree in computer science from the University of Utah in 1987. He is an associate professor of computer science at the University of Utah. From 1997 to 1999, he was a research associate professor in computer science at Utah. From 1989 to 1997, he was a technical staff member in the Advanced Computing Laboratory (ACL) located at Los Alamos

Peter Shirley received the BA degree in physics from Reed College and the PhD degree in computer science from the University of Illinois at Urbana-Champaign. He is an associate professor in the School of Computing at the University of Utah. He spent four years as an assistant professor at Indiana University and two years as a visiting assistant professor at the Cornell Program of Computer Graphics before moving to Utah. His professional interests

Allen McPherson holds the BS and MS degrees in computer science from Southern Illinois University and the University of New Mexico, respectively. He is a visualization researcher in the Advanced Computing Laboratory in the CCS Division at Los Alamos National Laboratory. His research interests include hardware-accelerated rendering, visualization and rendering algorithms, and cluster-based parallel computing. He is a member of the IEEE Computer Society.

► For more information on this or any computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.