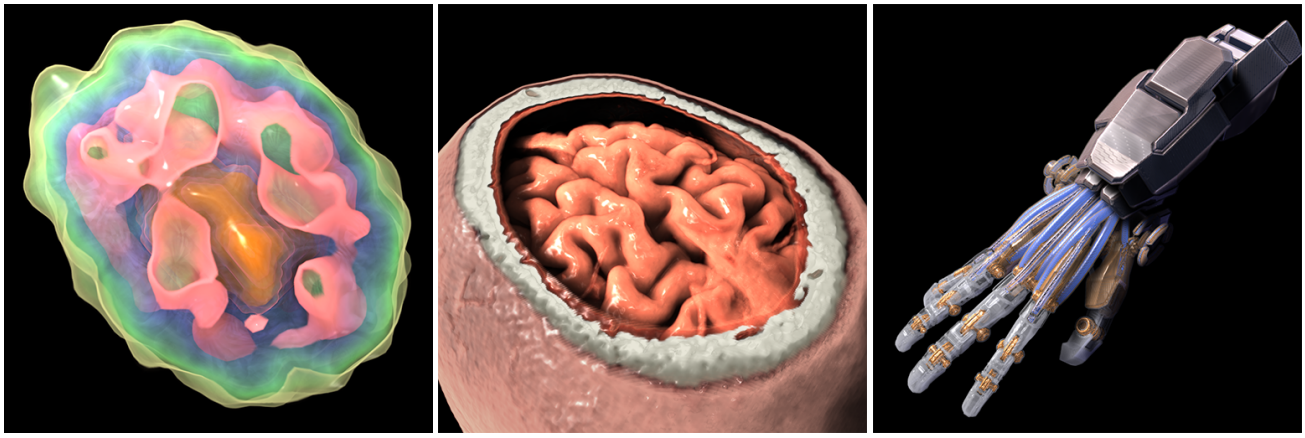
DOI: 10.1111/cgf.12916
Eurographics Conference on Visualization (EuroVis) 2016
K.-L. Ma, G. Santucci, and J. van Wijk
(Guest Editors)

Volume 35 (2016), Number 3

Decoupled Shading for Real-time Heterogeneous Volume Illumination

Y. Zhang[†] and K.-L. Ma[‡]

University of California Davis



(a) Cell, 47 FPS (b) Brain, 17 FPS (c) Mechanical Hand, 18 FPS

Figure 1: *Real-time rendering results of volume datasets with heterogeneous lighting and shading parameters.*

Abstract

Existing real-time volume rendering techniques which support global illumination are limited in modeling distinct realistic appearances for classified volume data, which is a desired capability in many fields of study for illustration and education. Directly extending the emission-absorption volume integral with heterogeneous material shading becomes unaffordable for real-time applications because the high-frequency view-dependent global lighting needs to be evaluated per sample along the volume integral. In this paper, we present a decoupled shading algorithm for multi-material volume rendering that separates global incident lighting evaluation from per-sample material shading under multiple light sources. We show how the incident lighting calculation can be optimized through a sparse volume integration method. The quality, performance and usefulness of our new multi-material volume rendering method is demonstrated through several examples.

Keywords: Global illumination, high performance, multi-material, multiple scattering, soft shadow, volume rendering

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism— I.3.8 [Computer Graphics]: Applications—

1. Introduction

One of the major goals in volume visualization is to accurately present the spatial structures of the volume data. A single volume dataset usually contain multiple features and they are classified into subsets, which are typically visualized using different

[†] ybzhang@ucdavis.edu
[‡] ma@cs.ucdavis.edu

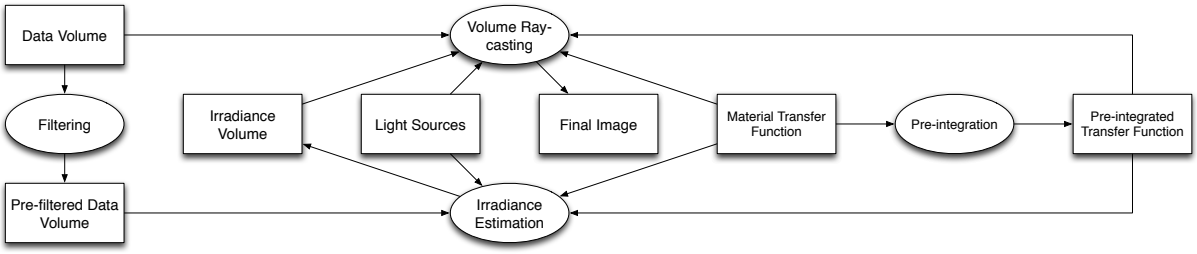


Figure 2: *The diagram of our rendering pipeline. Rectangular nodes are data objects and ellipsoidal nodes are operations.*

combinations of colors and opacity values. In order to achieve real-time frame rates for interactive volume exploration, local illumination with Phong shading [Pho75] is adopted in most interactive volume rendering systems. However, local illumination fails to present certain depth cues in the data such as occlusion, global lighting and shading. In recent studies, incorporating advanced lighting models into interactive volume rendering systems have been investigated extensively. In terms of computational cost, these models can be classified into short-range illumination and long-range illumination. Short-range illumination models evaluate light-material interactions within a short range, such as local directional occlusion [SPH*09]. Long-range illumination models compute light propagation in the entire volume domain, such as extinction-based shading and illumination [SMP11] and visibility-based global shadows [KJL*12]. Most previous studies on advanced volumetric lighting models focused on global light absorption and scattering. However, even with carefully designed transfer functions and lighting, it is not adequate to model heterogeneous volume illumination to achieve distinct visual appearances for different features in a volume dataset.

To explore the capabilities of rendering multi-material volume data in real-time, a complex shading model has to be incorporated into the classic emission-absorption volume integral. The main challenge comes from the evaluation of the high-frequency view-dependent lighting because the irradiance has to be calculated globally, which involves a secondary ray integration. While the low-frequency view-independent lighting can be evaluated per-voxel as most previous methods do, the view-dependent lighting needs per-sample shading along the primary ray. A straightforward implementation that evaluates the view-dependent component per-sample cannot achieve interactive frame rates. In this paper, we present a novel and practical technique that decouples the irradiance estimation from the material shading. While the materials are shaded per-sample, the global irradiance intensities are estimated per-light at sparsely distributed voxel centers and they are stored in an irradiance volume. It should be pointed out that we store the per-light irradiance instead of radiance such that the results can be reused with different view directions across multiple frames. As shown in Figure 1, our technique can simulate heterogeneous light-material interactions to generate realistic volumetric lighting and shading. We are not aware of similar interactive volume visualization systems with such capabilities.

2. Related Work

Direct volume rendering (DVR) involves approximating the solution of the radiative transfer equation [Cha60]. Max [Max95] reviewed several light transportation models commonly used in DVR. To reduce the sampling rate while maintaining the rendering quality, pre-integrated volume rendering is introduced [MHC90, SBM94]. It is also extended to support smooth lighting [LWM04] by linear interpolation between two weighted volume integrals.

Incorporating advanced lighting effects into DVR extends the capability of volume visualization systems. It has received much attention in recent years. Due to the high computational cost of global illumination, many introduced techniques are based on pre-computation and model simplification. Ropinski et al. [RMSD*08] introduced a pre-computation based technique to interactively render occlusion and color bleeding effects. To reduce the occlusion estimation time, summed area table (SAT) based techniques [DVND10, SMP11] are designed to efficiently look up the approximate occlusion information, but the table update is not real-time as the opacity changes. SAT is also used for accelerating shadow ray marching and shadow softness control [ASDW14]. Ritschel [Rit07] adopted a pre-computed radiance transfer (PRT) framework for DVR and proposed a fast radiance transfer re-computation technique for dynamic transfer function changes. Kronander et al. [KJL*12] introduced a visibility encoding technique using spherical harmonics (SH) for global shadow effects. Zhang and Ma [ZDM13] incorporated photon mapping and new basis functions into the PRT framework for volume rendering with high-frequency lighting. However, these methods also suffer from re-computation if the opacity is updated.

Without pre-computation or with parameter-independent pre-computation, advanced illumination can be achieved at a higher render-time cost. Qiu et al. [QXF*07] adopted a lattice-based global illumination method for volume rendering. Their method traces photons using lattices to fully simulate the light transport but it takes minutes for each tracing process. The screen-space ambient occlusion method [SA07], which renders local shadows by sampling nearby screen-space depth values, was extended to DVR [DVND10]. Schott et al. [SPH*09] uses cone-shaped phase function to model local directional occlusion. Ropinski et al. [RDRS10] presented a volume-space light simulation technique for shadow and single scattering. But it does not support interior light sources. Zhang and Ma [ZM13a] introduced a convection-diffusion model to approximate global illumination effects including global shadow and isotropic multiple scattering. Ament et al. [ASW13] presented an ambient volume scattering technique to approximate local anisotropic scattering effects using a pre-integrated table which does not depend on transfer functions. Both methods are limited to homogeneous volume materials.

To achieve different visual appearances on isosurfaces,

Stegmaier et al. [SSKE05] presented a volume rendering framework using single pass GPU ray-casting. They provided several GPU shaders to render different materials but these shaders cannot be combined together. Bruckner and Gröller [BG07] introduced style transfer functions for illustrative volume rendering which do not support global illumination. Lindemann and Ropinski [LR10] extended the PRT framework to support real-time volume rendering with reflecting and scattering materials assigned through the transfer function. It is limited to low-frequency lighting due to the SH projection. Kroes et al. [KPB12] introduced an interactive Monte Carlo ray-tracing technique which supports realistic physically-based lighting effects including flexible light sources and complex material representation. However, it usually takes more than one second to converge to a clear result and it is not easy to adapt to brick-based out-of-core rendering. Ament and Dachsbacher [AD16] introduce a structure-aligned specular lighting model for DVR to improve the perception of orientation and shape of surface-like structures. For an up-to-date review of previous volumetric illumination techniques, we refer readers to a recent survey [JSYR14].

3. Overview

3.1. Volume Illumination Model

According to [Cha60], the light transport equation has the form:

$$(\omega \cdot \nabla)L(x,\omega) = L_e(x,\omega) - \sigma_t(x,\omega)L(x,\omega) + \sigma_s(x,\omega)\int_\Omega L_i(x,\omega_i)p(x,\omega_i,\omega)d\omega_i \quad (1)$$

where $L(x,\omega)$ is the radiance at position x in direction ω, L_e is the radiance emitted from x in the same direction, σ_t is the attenuation coefficient, σ_s is the scattering coefficient, Ω is the unit sphere, L_i is the incident radiance from ω_i and p is the phase function which models the probability of light direction change from ω_i to ω. A full global illumination solution requires evaluating Equation 1 along all possible paths from the light sources to the camera. In volume ray-casting with simple local illumination models, Equation 1 is only evaluated along the primary rays, assuming there is no occlusion between the light sources and the primary rays. In the context of volume data visualization, most advanced rendering techniques focus on the partial or full evaluations of secondary rays with a single light bounce, which dominate the shadow and single scattering effects. The presented algorithm estimates the irradiance samples in the voxel-space based on the fully integrated secondary rays from all the light sources as well as multiple scattering. The visual appearance of the volume data is controlled by the phase function p together with the surface BRDF, which are referred to as material properties in this paper. It is discussed in Section 4.

3.2. The Rendering Pipeline

To approximate Equation 1, the volume rendering pipeline consists of four stages:

- **Material Editing** Given a volume dataset, the transfer function is created and edited by the user. The transfer function contains color, opacity and other material properties used in the rendering stages.
- **Pre-integration** Based on the transfer function, pre-integration tables are built for efficient ray integration and smooth surface lighting.
- **Light Simulation** The light simulation stage involves fast and sparse irradiance estimation in the voxel-space which takes material properties into account. The result is stored in a compact manner for final ray-casting.
- **Volume Ray-casting** At the ray-casting stage, the irradiance data are gathered along the primary rays to estimate the excitant radiance for final ray integration. Material properties are applied to the excitant radiance estimation.

Figure 2 shows a general diagram of the rendering pipeline. All the data objects and operations are discussed in the following sections.

4. Material Representation

4.1. Material Model and Parameters

For intuitive material editing, the material model combines the Blinn-Phong surface shading model [Bli77] and a simple volume light propagation model. The surface shading model has the form:

$$L(x,\omega) = \sum_{k=1}^{m} B(\omega,\omega_k,n(x),k_a(x),k_d(x),k_s(x),k_e(x))L_i(x,\omega_k) \quad (2)$$

where L is the excitant radiance, L_i is the incident radiance, m is the number of light sources, B is the Blinn-Phong shading function which depends on the view direction ω, the direction from the kth light source ω_k, the surface normal n, and the ambient k_a, diffuse k_d, specular k_s and shininess k_e coefficients. Here the ambient term is placed inside the summation which is different from the original definition. The intention is to make the ambient contribution also affected by the directional occlusion. For volume light propagation, it is assumed that there is no emission from the volume data:

$$(\omega \cdot \nabla)L(x,\omega) = -\sigma_t(x,\omega)L(x,\omega) + L_s(x,\omega,s) \quad (3)$$

where L is the radiance, σ_t is the attenuation coefficient, s is the scattering strength and L_s is the in-scattering term. In the implementation, the term L_s is approximated through diffusion. It is discussed in Section 5.2. In order to evaluate Equation 2 and Equation 3, the following parameter space is chosen:

c	material color
α	material opacity
k_a	ambient coefficient
k_d	diffuse coefficient
k_s	specular coefficient
k_e	shininess coefficient
σ	attenuation coefficient
s	scattering strength
r	reflection coefficient

Here σ and s are assumed to be isotropic for simplicity in material editing but there is no limitation on using anisotropic parameters in the rendering pipeline. Note that in real world the attenuation equals to the opacity $\sigma = \alpha$ but the proposed model use $\sigma = c_t\alpha$ which offers the flexibility to adjust the light transmittance in practice. c_t is a linear scaling parameter adjustable in the transfer function editor. A reflection parameter r is also added for environment mapping [BN76] which is useful for rendering metal or liquid-like materials.

4.2. Transfer Function Extension

According to Section 4.1, the transfer function is extended with extra parameters in addition to the color and opacity:

$$M : t \rightarrow (c, \alpha, k_a, k_d, k_s, k_e, \sigma, s, r) \quad (4)$$

All the material parameters are defined on a set of nodes which are editable by the user. Each node maps a single data value $t \in [0,1]$ to its corresponding material parameters. The material parameters between any two adjacent nodes in $[0,1]$ are linearly interpolated. Node-based material editing also allows us to build a material library with effect previews such that users can intuitively pick any desired materials without directly manipulating these parameters.

4.3. Transfer Function Pre-integration

Two pre-integration tables are built each time the color and opacity mappings are changed. The first table stores the pre-integrated color and opacity values within a fixed sample interval Δx:

$$I_c(t_1,t_2) = c_s \int_{t_1}^{t_2} c(t)\alpha(t) e^{-c_s \int_{t_1}^{t} \alpha(\tau) d\tau} dt \quad (5)$$

$$I_\alpha(t_1,t_2) = c_s \int_{t_1}^{t_2} \alpha(t) e^{-c_s \int_{t_1}^{t} \alpha(\tau) d\tau} dt \quad (6)$$

where $t_1, t_2 \in [0,1]$ are volume data values defined at the two endpoints x_1 and x_2, c is the color, α is the opacity and $c_s = \frac{\Delta x}{t_2 - t_1}$ is the scaling parameter from t to x. To use pre-integration, the basic assumption is that Δx is small and the scalar value being visualized varies linearly within Δx. The pre-integrated color and opacity values can be looked up during the integration process in volume ray-casting. Using color and opacity pre-integration helps eliminating color discontinuities when relatively large integration steps are used. However, it does not eliminate lighting discontinuities if the lighting is only evaluated at the endpoints of sample intervals because the intersection points of the rays and the isosurfaces are not aligned with the endpoints of sample intervals. To achieve smooth lighting, the second table stores the interpolation parameters:

$$I_\beta(t_1,t_2) = \beta \quad (7)$$

where β satisfies $0 \le \beta \le 1$, $\alpha(t_1 + \beta) > 0$ and $\alpha(t_1 + \beta') = 0$ for all $0 \le \beta' \le \beta - \varepsilon$, for some $\varepsilon > 0$. β is used as the parameter to interpolate the texture coordinates for normal and material sampling between the two endpoints of a sample interval. It is inspired by the pre-integration technique for high quality lighting [LWM04]. However, instead of shading at the two endpoints of each ray segment, a point x between the endpoints is chosen as the shading position:

$$x = (1-\beta)x_1 + \beta x_2 \quad (8)$$

where x_1 and x_2 are the endpoints. Therefore, the final shaded colors do not rely on the irradiance data from the regions where $\alpha = 0$. The advantage is that evaluating the irradiance in these empty regions can be skipped. Note that x does not necessarily coincide with the isosurface in the physical space since it is just a linear approximation in the transfer function space.

5. Light Simulation

Following Equation 2 and Equation 3, the light simulation consists of two stages. At the first stage, the irradiance is estimated in the voxel-space using Equation 3. The resulting irradiance data are then gathered at the second stage through volume ray-casting where Equation 2 is used for surface shading. The two-stage shading algorithm is based on the observation that the frequency of irradiance depends on the volume resolution rather than the sampling frequency of the volume ray-casting. In addition, the irradiance calculation within the empty regions can be avoided. Therefore the computational cost for secondary ray integration can be significantly reduced. The irradiance estimation stage is discussed in the following subsections.

5.1. Irradiance Estimation

The purpose of irradiance estimation is to simulate the global soft shadow and multiple scattering effects. Although fast pre-computation based algorithms can be adopted, such as spherical harmonics (SH) projection [LR10] or visibility encoding [KJL*12], these techniques are limited in dealing with high-frequency lighting due to the mathematical properties of the SH basis functions. Forward light propagation techniques [RDRS10, ZM13a] also have the limitation of simulating one light source at a time. To avoid these limitations, the irradiance estimation presented in this paper is based on direct ray-tracing from each sample point to all the light sources. Thus, high-frequency lighting can be achieved without any pre-computation. The irradiance estimation can also be implemented in a single pass GPU shader. The key to achieving real-time performance involves choosing sparse sample points and using large step for ray integration.

In general, the irradiance intensities are sparsely sampled at regular grid points where the opacity values are non-zero and the intensities are stored in an irradiance volume. This is compatible with the pre-integration schemes described in Section 4.3 since there is no need to shade in empty regions. At each sample point, m intensity values are stored, where m is the number of light sources. The light positions, directions, colors and other properties are stored in separate tables. The irradiance volume can have the same or lower resolution compared to the original dataset under the constraint that the whole volume fits in the GPU RAM. For example, a 128^3 irradiance volume with 4 light sources costs 8MB of GPU RAM. The volume renderer supports high-resolution data volumes, but it is adequate to use a lower resolution irradiance volume while maintaining similar shading quality, because the ray-integration is performed using a low-resolution pre-filtered data volume which avoids aliased shadows. Section 8 has further discussions about the irradiance volume resolution with examples.

It should be pointed out that the implementation of parallel calculations at sparse locations on GPU is non-trivial, which is discussed in Section 7. A naïve implementation may not benefit from the sparsity. After choosing the sparse sample points, the ray-integration is achieved by utilizing the pre-integrated opacity values discussed in 4.3. Given any volume dataset, a pre-filtered data volume which has the same resolution as the irradiance volume is first generated using trilinear filtering. At each sample point x, m rays are traced toward the m light sources until reaching the endpoint x_k where $1 \le k \le m$. The ray integration has the form:

$$I(x,\omega_k) = 1 - \int_x^{x_k} \sigma(x') T(x,x') dx' \quad (9)$$

where I is the portion of light that reaches x from the kth light source and T is the transmittance between two points:

$$T(x_a,x_b) = e^{-\int_{x_a}^{x_b} \sigma(x') dx'} \quad (10)$$

With the pre-integrated opacity table, large integration steps can be used to evaluate Equation 9, which is essential for real-time performance. Early ray termination is also adopted to accelerate the computation. The actual irradiance intensity is $I(x,\omega_k)c_k$ where c_k is the kth light color but they are stored separately to save space. The

time complexity of the irradiance estimation method is $O(MN^{4/3})$ where M is the number of lights and N is the number of sample points. Experiments show that the irradiance estimation step costs less time than the final ray-casting step, which is discussed in Section 9. The ray direction ω_k and the endpoint x_k in Equation 9 are determined by the type of the light source. The renderer supports three common types of light sources, including distant light, point light and spot light. They are discussed in the following subsections.

5.1.1. Distant Light

Distant lights are a kind of light source located at an infinite distance. They only have directional information. For any distant light, the ray direction ω_k in Equation 9 is set directly to the light direction d_k. The endpoint x_k is set to the point where the ray leaves the data volume. It is assumed that the distant light is parallel and there is no light attenuation in empty regions. The light direction is also used as the incident angle of the kth irradiance sample. The corresponding intensity is calculated through:

$$I_k(x) = I(x, d_k) \tag{11}$$

where I is the light visibility defined in Equation 9.

5.1.2. Point Light

Point lights have positions but no directional information. For any point light, the ray direction ω_k in Equation 9 is set to $x_{l,k} - x$ where $x_{l,k}$ is the light position and x is the sample point. The endpoint x_k is set to the light position $x_{l,k}$ if the light is inside the volume, or to the point where the ray leaves the data volume if the light is outside the volume. The point light has quadratic attenuation in empty regions. The direction $x - x_{l,k}$ is used as the incident angle of the irradiance sample. In the implementation, point light sources are approximated using tiny spherical sources. The corresponding intensity is calculated through:

$$I_k(x) = \frac{\varepsilon^2}{\max\left\{\varepsilon^2, \left\|x - x_{l,k}\right\|^2\right\}} I(x, x - x_{l,k}) \tag{12}$$

where ε is the radius of the point light source.

5.1.3. Spot Light

Spot lights are similar to point lights which have positions except that they also have angular information. A direction d_k and an angle ϕ_k are used to represent the visible cone region. For any spot light, the ray direction ω_k and the endpoint x_k are set in the same way as the point light. The spot light also has quadratic attenuation in empty regions. The direction $x - x_{l,k}$ is used as the incident angle of the irradiance sample. The difference is that Equation 9 is evaluated only if x is inside the visible cone region:

$$I_k(x) = \frac{\varepsilon^2}{\max\left\{\varepsilon^2, \left\|x - x_{l,k}\right\|^2\right\}} V(x, x_{l,k}, d_k, \phi_k) I(x, \omega_k) \tag{13}$$

where V is the visibility function:

$$V(x, x_l, d, \phi) = \begin{cases} 1, \frac{x - x_l}{\|x - x_l\|} \cdot d > \cos(\frac{\phi}{2}), \\ 0, otherwise. \end{cases} \tag{14}$$

5.2. Multiple Scattering

The irradiance estimation described in Section 5.1 only resolves the light attenuation in Equation 3. To approximate the in-scattering term, a diffusion equation is adopted:

$$\frac{\partial I_k}{\partial t} = \nabla \cdot (sD(I_k)\nabla I_k) \tag{15}$$

where s is the scattering strength and D is the diffusion tensor. Figure 3 is an illustration of volume scattering. The evaluation of radiance at the red point consists of two parts including single scattering and multiple scattering. Single scattering is the amount of light that travels from the light source to the red point directly with one bounce, which is evaluated through Equation 9 together with surface shading as discussed in later sections. Multiple scattering is the amount of light that bounces into the red point from other positions such as the green and blue points. Assuming that the distance between the red and green points is shorter than the distance between the red and blue points, there is a higher chance to have light scattered from the green point to the red point. Therefore, the diffusion equation can be a good approximation for such phenomena. In theory, the diffusion should be applied after evaluating the excitant radiance. But an early diffusion step can achieve similar effects with much less computational cost. It has to be evaluated only if the lights are changed, which is independent of the camera movement. Although D can be an arbitrary matrix or even a nonlinear matrix function of I_k, it is assumed that D is an identity matrix to maintain a small parameter space for material editing. By solving Equation 15 at $t = 1$ using $\Delta x = 1$, long-range multiple scattering effects can be achieved by using a large value of s.

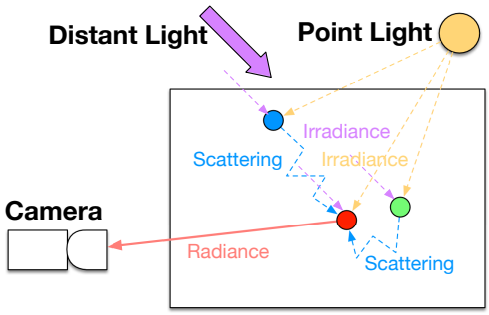
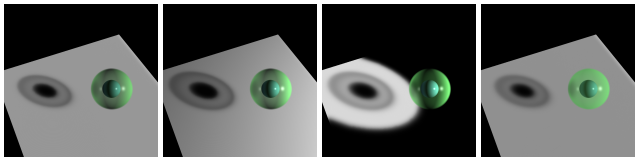


Figure 3: *Illustration of irradiance estimation and volume scattering. The radiance at the red point consists of both single scattering from the light sources directly and multiple scattering from other points such as the green and blue points. The irradiance is sparsely estimated through secondary ray-integration toward light sources and the scattering is approximated using volume-space diffusion. The pre-integrated irradiance is then used for radiance sampling at a higher frequency along the primary rays.*



(a) Distant (b) Point (c) Spot (d) Scattering

Figure 4: *Illustration of using different light sources and the multiple scattering effect on the green sphere. All the examples run at 49 FPS. The performance does not depend on the type of light source because it is memory bandwidth limited, not math limited.*

6. Volume Ray-casting

The volume renderer supports brick-based out-of-core GPU rendering for large volume datasets. One of the challenges is integrating the advanced material shading techniques into such rendering system. The algorithm used for brick-based rendering is first described in the following subsection.

6.1. Brick Composition

Brick-based volume rendering (e.g. [BHMF08]) is crucial in modern volume visualization systems since large volume datasets are common. These datasets typically do not fit in the GPU RAM and out-of-core rendering is required. Given any volume dataset, it is divided into cubical bricks. The maximum and minimum data values of each brick are pre-calculated. Before volume-ray casting, the renderer compares the brick data ranges to the summed area table of the opacity transfer function and filter out all the empty bricks. The remaining bricks are sorted in front-to-back order. With limited GPU RAM, these bricks are sent to the GPU in batches and are cached in 3D volume textures. Then the volume ray-casting algorithm is performed on each brick in order and the results are composited using alpha blending.

For each brick, two extra layers (a.k.a. ghost cells) which overlap with neighboring bricks are kept for both data value interpolation and smooth gradient calculation. In order to utilize 2D pre-integrated transfer functions, a uniform sample interval is used and the ray segments are globally aligned. The aligned ray segments can partially overlap with the brick but are fully covered by the extra layers.

6.2. Surface Shading

To incorporate advanced lighting-material interaction, the irradiance volume is kept in the GPU RAM in addition to the currently cached data bricks. By limiting the size of the irradiance volume, the irradiance data can always be stored in the GPU RAM. During the volume ray-casting, each ray is divided into uniform ray segments. Given a ray segment with endpoints x_1 and x_2, the corresponding data values $t_1 = t(x_1)$ and $t_2 = t(x_2)$ are first sampled. Then the pre-integrated color $c = I_c(t_1,t_2)$, the opacity $\alpha = I_\alpha(t_1,t_2)$ and the interpolation parameter $\beta = I_\beta(t_1,t_2)$ can be obtained. The point x to be shaded is calculated through Equation 8. The material transfer function M defined in Equation 4 and the irradiance intensities I_k are sampled at x. The surface normal n is also sampled at x using the normalized gradient of the volume data, which is evaluated through the central difference. Together with the camera ray direction ω and local irradiance direction ω_k where $1 \le k \le m$, the following definition of the shading function B from Equation 2 can be derived:

$$B(M) = [k_a + k_d(n\cdot\omega_k)]\,c(x)\alpha(x) + k_s\left(n\cdot\frac{\omega-\omega_k}{\|\omega-\omega_k\|}\right)^{k_e}\alpha(x) \tag{16}$$

The incident radiance can be computed through $L_i(x,\omega_k) = I_k(x)c_k$ where c_k is the light color and I_k is defined in Section 5.1. By plugging Equation 16 into Equation 2, the surface shading can be fully evaluated. In fact, B can be replaced with any other shading model which can be evaluated in constant time. The Blinn-Phong shading model is chosen because it has intuitive shading parameters which are easy to manipulate.

The specular reflection effect can also be added by modifying Equation 2:

$$L(x,\omega) = \sum_{k=1}^{m} B(M)I_k(x)c_k + r(x)I(x,\omega_r)c_r(\omega_r) \tag{17}$$

where $\omega_r = \omega - 2(n\cdot\omega)n$ is the direction of reflection, r is the reflection parameter, I is the visibility in direction ω_r and c_r is the environment color in direction ω_r. Reflection is applied to rendering metal or liquid-like materials. The environment colors are stored in cube map textures.

7. Implementation

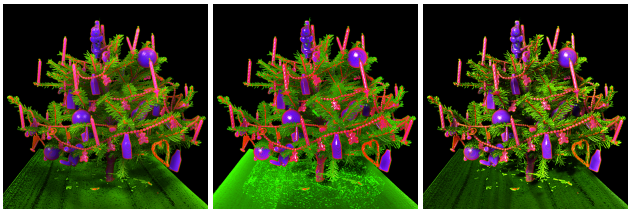
Based on the previous discussion, the general procedure is listed in Algorithm 1. We implemented the whole algorithm in a Qt and OpenGL based software application. In the implementation, it is assumed that the data size can be larger than the available GPU RAM. During the loading process, the volume dataset is divided into cubical bricks where the brick size can be configured by the user. The brick size is set to 64^3 in all the experiments presented in Section 8. Data pre-filtering is performed via a trilinear filtering which is similar to the 2D mipmap technique. Two mip-levels that are closest to the irradiance volume resolution are chosen for the filtering. To store irradiance intensities, multiple 3D volume textures with 32-bit RGBA format are used where each channel stores an 8-bit intensity value. Each texture can store the irradiance intensities for up to 4 light sources. In most cases, only a single 3D texture is needed as the number of lights is less than 4. A single pass GLSL shader is implemented to compute the irradiance volume. Multiple render targets (MRT) are used if there are two or more 3D textures. Although it is possible to store radiance directly, there are several advantages to store irradiance:

- The irradiance intensities can be reused with multiple materials which have different reflective properties.
- The irradiance intensities can be reused with different view directions, which enables faster rendering of multiple views.
- The irradiance intensities can be reused across multiple frames with dynamic view directions and light colors as long as the relative positions between the light sources and the volume are unchanged.

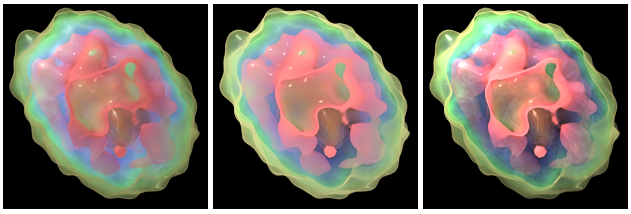
With pre-integrated opacity, large integration steps and early ray termination are used to achieve fast and accurate irradiance estimation. In addition to large integration steps, another important technique which contribute to rendering performance is coalescing the calculations at sparse voxel locations. Since the calculations are performed layer by layer, we use an additional pass to generate the sparse stencil mask before the irradiance estimation for each layer. With the stencil mask, GPU can generate right amount of threads without any divergence so all the stream processors can be fully utilized, which is crucial for real-time rendering. The scattering is implemented in a diffusion shader which also takes multiple 3D textures as input and output. The transfer function pre-integration and the final volume ray-casting are implemented in separate GLSL shaders.

Algorithm 1 Multi-material Volume Rendering

```
Prepare original volume dataset
Create pre-filtered data volume in GPU
Create irradiance volume in GPU
Prepare lighting and material transfer function M(t)
if M(t) is changed then
    Pre-integrate c, α and β
if Lighting or M(t) is changed then
    Estimate irradiance intensity I_k(x)
    Apply scattering to I_k(x)
if Camera is changed and reflection is enabled then
    Estimate I(x, ω_r)
if Camera, lighting or M(t) is changed then
    Clear frame buffer
    Sort bricks in front-to-back order
    for each data brick do
        for each pixel or sample do
            for each ray segment do
                Evaluate L(x, ω)
                Integrate L(x, ω) to final color
        Blend result to frame buffer
Output frame buffer
```



(a) AO (b) AVS (c) Ours



(d) AO (e) AVS (f) Ours

Figure 5: *Comparison between previous volume rendering methods including local ambient occlusion (AO), ambient volume scattering (AVS) and our multi-material method using the Christmas tree dataset and the keratinocyte cell dataset. It is shown that previous methods are limited to homogeneous materials. By using our method, the Christmas tree leaves and decorations, the cell organelle and cytosol can have their own materials and present different looks.*

8. Examples

In this section, several examples are used to demonstrate our heterogeneous volume illumination technique. The transfer function and lighting design are discussed before these examples.

8.1. Transfer Function and Lighting Design

With multi-material shading capability, users need to provide the material descriptions for all features within a volume dataset. In our experiments, we start with placing one material node per individual feature with Gaussian opacity distribution. Each node is associated with a set of intuitive material parameters described in Section 4. Users can visually edit each material node to see how it affects the final rendering. The transfer function creation process for the mechanical hand in Figure 1(c) is presented in the supplementary video. The edited material node can be reused such that expert users may create a set of pre-defined materials and normal users can apply them directly. In practice, we also limit the number of principal colors to be less than four, which leads to satisfactory visualization results. We did a preliminary experiment on the average time spent on transfer function editing. The results are shown in Table 1. For lighting design, our rendering system can automatically generate the key light and the fill light similar to the three-point lighting design strategy [ZM13b] except that the back light is not used. Users can modify the generated light sources based on special requirements.

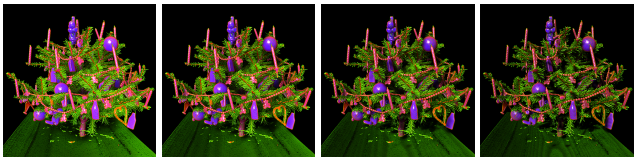
#Materials	Expert (sec.)	Normal (sec.)
2	51/20	143/35
3	87/37	271/78
4	145/47	431/112

Table 1: *Average time spent on transfer function editing from 5 expert users (with volume rendering experience) and 10 normal users. The users are asked to edit the transfer functions to match given visualization examples. The timing data are measured twice where template material nodes are provided in the second measurement.*

8.2. Christmas Tree

A scanned Christmas tree dataset is used as the first example. This dataset mainly consists of leaves and other decorations. One of the challenges to render such volume data is to differentiate the leaves and other decorations. As shown in Figure 5(a), standard volume ray-casting with local ambient occlusion is not adequate to present the leaves with correct shading. The object-space ambient occlusion fails to capture global high-frequency shading effects under strong directional lights. The ambient volume scattering technique [ASW13] is also applied to the dataset shown in Figure 5(b). The global shading effects can be captured due to the SAT-based occlusion but the leaves look unrealistic due to its homogeneous material parameters and scattering radius. With the presented irradiance estimation and heterogeneous material shading technique, realistic surface shading and scattering effects can be reproduced with ease, as shown in Figure 5(c). In addition to the Phong shading parameters, a high attenuation parameter $\sigma = 1$ is used for the leaf material to capture high-frequency details. A scattering parameter $s = 4$ is used for short-range scattering. For the decoration materials, smaller attenuation $\sigma \leq 0.5$ and stronger scattering $10 \leq s \leq 20$ are used to avoid overshadowing. The rendering results show that the multi-material technique has clear advantages over the state of the art volume rendering techniques on both shading quality and realism. While previous real-time volume rendering techniques with advanced lighting can also produce global shadows and scattering, the presented method focuses more on the realistic light-material interaction. Such capability can help users to understand the dataset in an intuitive way and also generate better visualizations for illustration and presentation.

To study the shading quality, the Christmas tree dataset is ren-



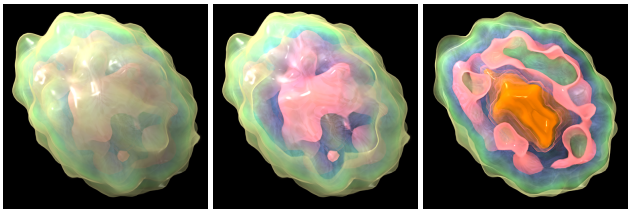
(a) 32^3, 6ms (b) 64^3, 9ms (c) 128^3, 16ms (d) 256^3, 49ms

Figure 6: *Rendering of the Christmas tree dataset using different irradiance volume resolutions. Using a high resolution irradiance volume can capture high-frequency shadow while using a low resolution irradiance volume leads to softer shadow.*

dered using different irradiance volume resolutions. The results are listed in Figure 6. It is shown that using low resolution irradiance volumes can only generate low-frequency soft shadows while using high resolution irradiance volumes can capture high-frequency shadows. Obviously, the computational cost increases as the irradiance volume resolution increases. The corresponding irradiance estimation performance is also measured. In practice, the shadow frequency and the usage of GPU RAM should be balanced. According to the experiments, we found that a 64MB irradiance volume is sufficient for most cases including high resolution datasets.

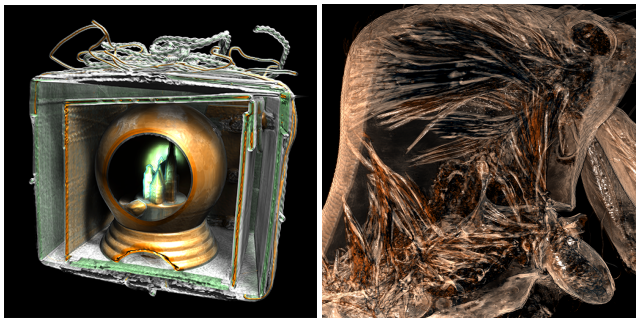
8.3. Keratinocyte Cell

A segmented keratinocyte cell dataset is also used to demonstrate the multi-material volume rendering technique. The rendering results are listed in Figure 7. While Figure 7(a) shows the whole cell, Figure 7(b)-(d) show its cutaway views with different clipping planes to reveal the internal structures. Soft tissue like materials are assigned to the organelle. These materials have high specular coefficients $0.8 \leq k_s \leq 1.0$ and high scattering coefficients $10 \leq s \leq 20$. Liquid like reflective materials with $0.1 \leq r \leq 0.5$ and low opacity are assigned to the cytosol. In this example, it is shown that the presented technique can generate high quality illustrative visualizations with advanced material shading. We also compared the results with previous methods in Figure 5. While previous methods cannot present these materials effectively, the presented method can differentiate the materials through heterogeneous light-material interaction because the rendering algorithm allows objects with the same opacity value having different light transmittance and scattering coefficients.



(a) Clip $x_{\max} = 1.0$ (b) Clip $x_{\max} = 0.8$ (c) Clip $x_{\max} = 0.65$

Figure 7: *Rendering of the keratinocyte cell dataset. Different clipping planes are used to reveal the interior of the cell. Different materials are used to illustrate the organelles of the cell.*



(a) (b)

Figure 8: *(a) Rendering of the present box dataset in a cutaway view. In addition to a directional light source, a spot light is also added to highlight the interior objects. (b) Rendering of a high resolution scanned sea spider in a cutaway view.*

8.4. Other Examples

The third example is a present box dataset with nested boxes and internal objects. In this example, the capability of rendering reflective materials are shown using a cutaway view of the present box. In Figure 8(a), reflective materials are used for the interior objects to simulate metal-like material. A spot light pointing to the center is also added to highlight the central part. Figure 8(b) shows the rendering results of a high resolution ($957 \times 1195 \times 1003$) scanned sea spider. This dataset is used for performance stress test. The timing results are presented in Section 9.

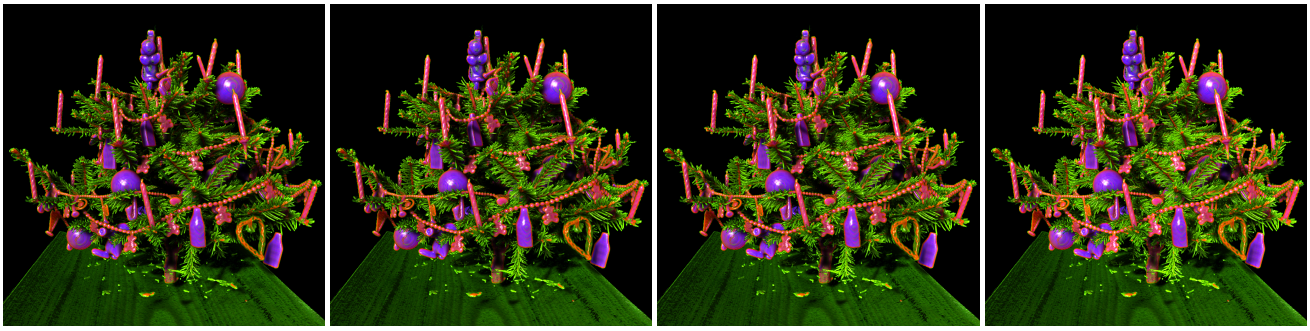
More examples are shown in Figure 1. In Figure 1(b), a scanned human brain dataset is rendered using soft-look materials consists of multi-layer translucent and highly specular surfaces. In Figure 1(c), a mechanical hand dataset is rendered using multiple plastic and metal materials. The proposed multi-material shading technique allows presenting distinct materials simultaneously within the real-time volume ray-casting framework. Such illumination effects cannot be achieved using any previous real-time volume rendering techniques.

8.5. User Evaluation

We designed a preliminary user evaluation through rendering different types of volume datasets using previous volume rendering methods including (a) precomputed photon mapping [ZDM13], (b) low-pass filtered shadow [ASDW14] and (c) our method. 19 users were asked to select the preferred rendered image for each dataset. We have picked 3 datasets from CT scans with sharp surfaces and 3 datasets from fluid simulations without clear surface features. Each dataset has at least 2 distinct features. For all datasets from CT scans, (a) was selected by 15.8% of the users; (b) was selected by 10.5% of the users; (c) was selected by 73.7% of the users. For all datasets from fluid simulations, (a) was selected by 28.1% of the users; (b) was selected by 35.1% of the users; (c) was selected by 36.8% of the users. The study shows that our method has clear advantages for visualizing volume datasets with sharp surfaces and multiple features.

Dataset Name	Data Res.	IV Res.	Lights	PI (ms)	IE (ms)	RC (ms)	FPS(AO/AVS/Ours)
Christmas Tree	$512 \times 499 \times 512$	$256 \times 250 \times 256$	1	1	91/49	125	4/2/6
Keratinocyte Cell	$74 \times 81 \times 43$	$74 \times 81 \times 43$	2	1	11/9	11	31/5/47
Present Box	$492 \times 492 \times 442$	$123 \times 123 \times 111$	2	1	24/15	97	6/2/9
Human Brain	$256 \times 256 \times 156$	$256 \times 256 \times 156$	2	1	32/22	23	12/4/17
Mechanical Hand	$640 \times 220 \times 229$	$320 \times 110 \times 115$	2	1	31/20	33	13/3/18
Sea Spider	$957 \times 1195 \times 1003$	$240 \times 299 \times 251$	2	1	38/29	313	2/1/3

Table 2: *Performance evaluation of the multi-material renderer on a MacBook Air with 1.3 GHz Intel Core i5 CPU (8GB RAM) with integrated HD Graphics 5000 GPU. Columns left to right: dataset name, data resolution, irradiance volume (IV) resolution, number of lights, pre-integration time (PI), irradiance estimation time (IE) without and with stencil mask optimization, ray-casting time (RC), FPS of ambient occlusion, ambient volume scattering and the multi-material method.*



(a) Reference,1582ms (b) $0.5\Delta x$,175ms,pre-integration (c) $1.0\Delta x$,153ms,pre-integration (d) $0.5\Delta x$,171ms,no pre-integration

Figure 9: *Quality and performance evaluation with different integration steps. Δx is the voxel size. The reference image is generated using per-sample irradiance estimation and $0.125\Delta x$. Other images are generated using our decoupled lighting and material shading technique.*

9. Performance

The performance of the presented rendering technique is measured for all the datasets used in this paper. The test machine is a laptop which has 1.3 GHz Intel Core i5 CPU with 8GB RAM and integrated Intel HD Graphics 5000 GPU. The average time cost for each rendering stage is listed in Table 2. For irradiance estimation, we listed both computation times without and with stencil mask optimization. The FPS data are also collected by using different rendering methods including ambient occlusion, ambient volume scattering [ASW13] and our method. The FPS data are collected with continuously changing transfer functions and light sources because users edit transfer functions frequently in the volume visualization process. Results show that the presented method has good performance on low-end machines and does not suffer from significant performance drop if users are editing the transfer function in real-time. Real-time rendering performance can be achieved for normal resolution ($\leq 512^3$) datasets. The multi-material method is slightly faster than the ambient occlusion (AO) method in these test cases because computing AO in object-space requires more samples than irradiance estimation if the number of light sources is small. The ambient volume scattering method requires rebuilding SAT data as the transfer function changes, which is time consuming and affects the overall performance. If the transfer function and light sources are fixed, the presented method also uses the least number of samples (i.e. texture fetch operations) per ray-segment during the volume ray-casting. Only two additional texture fetches are required including irradiance volume sampling and material transfer function sampling.

In addition, we evaluated the performance and quality of our decoupled lighting and material shading with pre-integrated opacity for irradiance estimation. Figure 9 shows that with much less computational cost, our results in 9(b) and 9(c) are close to the ground truth in 9(a). The shadow details are missing in 9(d) without pre-integration for irradiance estimation.

One of the limitations of our method is that global reflection and refraction are not supported due to the brick-based rendering system. Such effects can only be achieved by global ray-tracing which requires the entire volume dataset present in the GPU RAM. They are traded off to support large volume datasets since the latter is more crucial in volume visualization. Another limitation is that the fast irradiance estimation relies on the pre-integrated opacity table. Thus, only 1D transfer functions are supported which is not good enough for data classification in many cases. A multi-dimensional transfer function extension is possible.

10. Conclusion

A novel volume rendering framework for multi-material volume data visualization is presented in this paper. Results show that our renderer can generate high-quality illustrative or realistic images with advanced lighting in real-time. The presented lighting and shading techniques can also be integrated into other advanced volume rendering systems. In addition, the method does not require pre-computation and none of the parameters are limited to fixed ranges. The irradiance volume based rendering also has the flexibility to use even more general material representations such as general BRDFs and anisotropic phase functions.

The presented technique also has limitations in applicable volume visualization cases. The best use cases of our technique are visualizing volume data which has detectable sharp surface features with inner-homogeneity such that the integrated surface shading model can be utilized. In addition, the presented isotropic scattering model may not precisely depict the thickness of certain microstructures in biomedical applications.

Acknowledgments

This research has been sponsored in part by the U.S. Department of Energy through grant DE-SC0007443 and U.S. National Science Foundation through grants IIS-1528203 and IIS-1320229. Thanks go to Professor Sheng-Lung Huang at National Taiwan University and Dr. Dula Parkinson at the Lawrence Berkeley National Laboratory for providing the test datasets.

References

[AD16] AMENT M., DACHSBACHER C.: Anisotropic ambient volume shading. *IEEE Trans. Vis. Comput. Graph. 22*, 1 (2016), 1015–1024. 3

[ASDW14] AMENT M., SADLO F., DACHSBACHER C., WEISKOPF D.: Low-pass filtered volumetric shadows. *IEEE Trans. Vis. Comput. Graph. 20*, 12 (2014), 2437–2446. 2, 8

[ASW13] AMENT M., SADLO F., WEISKOPF D.: Ambient volume scattering. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2936–2945. 2, 7, 9

[BG07] BRUCKNER S., GRÖLLER M. E.: Style transfer functions for illustrative volume rendering. *Comput. Graph. Forum 26*, 3 (2007), 715–724. 3

[BHMF08] BEYER J., HADWIGER M., MÖLLER T., FRITZ L.: Smooth mixed-resolution gpu volume rendering. In *Proceedings of SPBG* (2008), pp. 163–170. 6

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph. 11*, 2 (July 1977), 192–198. 3

[BN76] BLINN J. F., NEWELL M. E.: Texture and reflection in computer generated images. *Commun. ACM 19*, 10 (Oct. 1976), 542–547. 3

[Cha60] CHANDRASEKHAR S.: *Radiative Transfer*. Dover Publications, New York, 1960. 2, 3

[DVND10] DÍAZ J., VÁZQUEZ P., NAVAZO I., DUGUET F.: Real-time ambient occlusion and halos with summed area tables. *Computers & Graphics 34*, 4 (2010), 337–350. 2

[JSYR14] JÖNSSON D., SUNDÉN E., YNNERMAN A., ROPINSKI T.: A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering. *Computer Graphics Forum 33*, 1 (2014), 27–51. 3

[KJL*12] KRONANDER J., JONSSON D., LOW J., LJUNG P., YNNERMAN A., UNGER J.: Efficient visibility encoding for dynamic illumination in direct volume rendering. *Visualization and Computer Graphics, IEEE Transactions on 18*, 3 (March 2012), 447–462. 2, 4

[KPB12] KROES T., POST F. H., BOTHA C. P.: Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE 7*, 7 (07 2012), e38586. 3

[LR10] LINDEMANN F., ROPINSKI T.: Advanced light material interaction for direct volume rendering. In *Proceedings of Volume Graphics* (2010), pp. 101–108. 3, 4

[LWM04] LUM E. B., WILSON B., MA K.-L.: High-quality lighting and efficient pre-integration for volume rendering. In *Proceedings of VISSYM* (2004), pp. 25–34. 2, 4

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph. 1* (1995), 99–108. 2

[MHC90] MAX N., HANRAHAN P., CRAWFIS R.: Area and volume coherence for efficient visualization of 3d scalar functions. *SIGGRAPH Comput. Graph. 24*, 5 (Nov. 1990), 27–33. 2

[Pho75] PHONG B. T.: Illumination for computer generated pictures. *Commun. ACM 18*, 6 (June 1975), 311–317. 2

[QXF*07] QIU F., XU F., FAN Z., NEOPHYTOS N., KAUFMAN A., MUELLER K.: Lattice-based volumetric global illumination. *IEEE Trans. Vis. Comput. Graph. 13*, 6 (2007), 1576–1583. 2

[RDRS10] ROPINSKI T., DÖRING C., REZK-SALAMA C.: Interactive volumetric lighting simulating scattering and shadowing. In *PacificVis* (2010), IEEE, pp. 169–176. 2, 4

[Rit07] RITSCHEL T.: Fast gpu-based visibility computation for natural illumination of volume data sets. In *Proc. of Eurographics* (2007), pp. 17–20. 2

[RMSD*08] ROPINSKI T., MEYER-SPRADOW J., DIEPENBROCK S., MENSMANN J., HINRICHS K.: Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum 27*, 2 (2008), 567–576. 2

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on gpus. In *Proc. of I3D* (2007), pp. 73–80. 2

[SBM94] STEIN C. M., BECKER B. G., MAX N. L.: Sorting and hardware assisted rendering for volume visualization. In *Proceedings of VVS* (1994), pp. 83–89. 2

[SMP11] SCHLEGEL P., MAKHINYA M., PAJAROLA R.: Extinction-based shading and illumination in gpu volume ray-casting. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (Dec. 2011), 1795–1802. 2

[SPH*09] SCHOTT M., PEGORARO V., HANSEN C., BOULANGER K., BOUATOUCH K.: A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum 28*, 3 (2009), 855–862. 2

[SSKE05] STEGMAIER S., STRENGERT M., KLEIN T., ERTL T.: A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *Proceedings of Volume Graphics* (2005), pp. 187–195. 3

[ZDM13] ZHANG Y., DONG Z., MA K.-L.: Real-time volume rendering in dynamic lighting environments using precomputed photon mapping. *IEEE Transactions on Visualization and Computer Graphics 19*, 8 (Aug. 2013), 1317–1330. 2, 8

[ZM13a] ZHANG Y., MA K.-L.: Fast global illumination for interactive volume visualization. In *Proc. of I3D* (2013). 2, 4

[ZM13b] ZHANG Y., MA K.-L.: Lighting design for globally illuminated volume rendering. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2946–2955. 7