IEEE/EG International Symposium on Volume Graphics (2010)
R. Westermann and G. Kindlmann (Editors)

Advanced Light Material Interaction for Direct Volume Rendering

Florian Lindemann and Timo Ropinski

Visualization and Computer Graphics Research Group (VisCG),
Department of Computer Science, University of Münster, Germany

Abstract

In this paper we present a heuristic approach for simulating advanced light material interactions in the context of interactive volume rendering. In contrast to previous work, we are able to incorporate complex material functions, which allow to simulate reflectance and scattering. We exploit a common representation of these material properties based on spherical harmonic basis functions, to combine the achieved reflectance and scattering effects with natural lighting conditions, i. e., incorporating colored area light sources. To achieve these goals, we introduce a modified SH projection technique, which is not just tailored at a single material category, but adapts to the present material. Thus, reflecting and scattering materials as assigned trough the transfer function can be captured in a unified approach. We will describe the required extensions to the standard volume rendering integral and present an approximation which allows to realize the material effects in order to achieve interactive frame rates. By exploiting a combination of CPU and GPU processing, we are able to modify material properties and can change the illumination conditions interactively. We will demonstrate the outcome of the proposed approach based on renderings of real-world data sets and report the achieved computation times.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

In recent years sophisticated volumetric illumination models for direct volume rendering (DVR) have been proposed. Interactive techniques are available, that support ambient occlusion [RMSD*08, SPH*09, LHY09], shadows [HKSB06] as well as scattering [KPH*03, Sal07]. While all these techniques brought realism in the are of volume graphics to a new level, an essential part needed to allow even more realistic image synthesis has only attracted little attention in the area of DVR so far: light material interaction. The appearance of an illuminated object depends on the combination of material properties, lighting properties as well as the light transport. While these three factors are often considered when illuminating polygonal objects, volume rendering techniques so far mainly focus on modeling the light transport. Regarding the material and the lighting properties only little work has been done, i. e., materials are usually reduced to assigning an emissive color through the transfer function and a point light source is used to illuminate the scene. However, only by considering complex reflectance functions in combination with area light sources, realistic illumination effects can be simulated. By not considering advanced material properties as well as area light sources, the quality of DVR images is rather limited compared to polygonal computer graphics.

Besides the material properties, light material interactions depend on the light source geometry, i. e., its extend and direction, and in some cases also the viewing direction. Therefore, changing the camera or the light source requires a recomputation. In this paper we introduce an approach for integrating different material types into the volume rendering process and illuminate them using area light sources. With the presented approach, it becomes possible to simulate realistic materials of different kinds interactively. We will demonstrate how to incorporate complex reflectance and phase functions as well as area light sources. By transforming all these properties into a common basis representation, we are able to perform all computations required for render-

ing on the GPU, and are able to achieve interactive frame rates, by still allowing to change the light properties and to a certain degree also the material properties. The key idea is to exploit a spherical harmonic (SH) basis function representation for the material properties as well as the light sources. This allows us to integrate colored area light sources, and simulate their interaction with advanced materials interactively, i. e., the light direction as well as the camera position can be changed. To our knowledge, this is the first attempt, which incorporates colored area light sources into DVR. Furthermore, we are not aware of other techniques dealing with advanced material properties in the area of interactive volume rendering, in the sense that complex reflectance functions as well as scattering functions can be used. Additionally, we exploit an adapted SH projection approach, which allows to handle different material categories, while allowing an easy integration of conventional SH lighting [SKS02].

2. Related Work

In the past, several interactive volumetric illumination models have been proposed. Many of these techniques are based on the optical models initially derived by Max [Max95]. One important approach in this area is the scattering technique presented by Kniss et al. [KPH*03]. By restricting themselves to a forward scattering phase function, the authors are able to achieve convincing results at interactive frame rates. The idea of constraining the phase function has been picked up by other researchers. Schott et al. extend this idea in order to integrate directional occlusion effects into a slice-based volume renderer [SPH*09]. Patel et al. have adapted the concept to not only incorporate occlusion information, but to also simulate light emitting structures [PBVG10]. However, since the techniques by Schott et al. as well as Patel et al. are based on a slice-wise front-to-back compositing, only headlights are possible. Ropinski et al. also use cone-shaped phase functions in order to simulate scattering effects together with an interactive volume processing technique, were light is swept through an illumination volume [RDRS10]. While all these techniques are capable to produce convincing results, they are rather strong approximations, since scattering is assumed to be directed along one major axis, and not over the whole sphere. This is especially problematic, when dealing with area light sources having multiple spatial components. An alternative approach has been introduced by Rezk-Salama [Sal07]. He proposes the use of Monte-Carlo integration to allow scattering effects on a limited set of surfaces.

Besides the integration of scattering effects, researchers have also targeted the simulation of diffuse interreflections. As with the related work regarding scattering, we cover only those publications targeting interactive volume rendering. Ropinski et al. use a set of local histograms in order to capture the vicinity of a voxel [RMSD*08]. While their technique is based on a rather expensive histogram precomputation, Ljung et al. exploit a ray-casting approach to determine the ambient illumination [LHY09]. Both of these approaches can be applied to DVR as well as isosurface rendering. In contrast, Wyman et al. [WPHS06] and Beason et al. [BGB*06] focus on a precomputation, which allows to render isosurfaces under static illumination conditions. More recently, Banks and Beacon have demonstrated how to decouple illumination sampling from isosurface generation, in order to achieve sophisticated isosurface illumination results [BB09].

SH basis functions have been used in many areas of polygonal rendering. They allow to approximate spherical functions by exploiting a set of SH basis functions. Among other applications, this concept has been exploited to capture BRDFs [WAT92], as well as to allow the integration of low frequency shadowing effects [SKS02]. It could also be shown, that SH basis functions can be used for other rendering effects, as caustics or scattering [KH84]. Ritschel has applied SHs in the field of volume rendering to simulate low-frequency shadowing [Rit07]. Similar as in this paper, a GPU-based approach is exploited in order to compute the required SH coefficients. The ray traversal with increasing step size as proposed by Ritschel, has also been integrated into our implementation.

As can be seen, all cited volume rendering papers rather target light transport, than light material interaction. However, we believe that this area needs to be covered in order to generate more convincing renderings. To our knowledge only the style transfer function approach for illustrative visualization by Bruckner and Groeller [BG07] deals with advanced material properties in the area of interactive volume rendering so far.

3. Light Material Interaction

In the following, we will incorporate advanced material properties as well as colored area light sources into interactive volume rendering. We will refer to reflectance and phase functions as material functions, and will call materials having significant differences as belonging to other material categories. When considering material and lighting properties together with light transport, the radiance $L(\vec{x},\vec{\omega}_o)$ which leaves from position $\vec{x}$ inside a volume in direction $\vec{\omega}_o$ can be defined as:

$$L(\vec{x},\vec{\omega}_o) = E_M(\vec{x},\vec{\omega}_o) + \int_\Omega M(\vec{x},\vec{\omega}_i,\vec{\omega}_o)\cdot L_i(\vec{x},\vec{\omega}_i)d\vec{\omega}_i.$$

Here, $L(\vec{x},\vec{\omega}_o)$ is the sum of the emission $E_M(\vec{x},\vec{\omega}_o)$ and the local light material interaction, which is expressed by the integral over the unit sphere Ω. To compute the local light material interaction, the material function $M(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ is modulated by the incident intensity $L_i(\vec{x},\vec{\omega}_i)$ coming from direction $\vec{\omega}_i$. Since we deal with volume data, where the

notion of a surface does not inherently exist, we do not weight this modulation by the clamped cosine term of the surface normal and ω_i. While this weighting could be easily achieved by using the gradient $\bigtriangledown\tau(f(\vec{x}))$, it would not compromise with the definition of the material function $M(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$, which is in contrast to surface reflectance functions not defined over the hemisphere, but over the whole sphere. So far, only local effects, such as emission and local light material interactions, i. e., out-scattering or reflectance, have been incorporated. To integrate environmental effects, as in-scattering or color bleeding, the unmodified incident intensity $L_i(\vec{x},\vec{\omega}_i)$ has to be modulated with $L_{mi}(\vec{x},\vec{\omega}_i)$, which is based on the environment of $\vec{x}$:

$$L(\vec{x},\vec{\omega}_o) = E_M(\vec{x},\vec{\omega}_o) + \int_\Omega M(\vec{x},\vec{\omega}_i,\vec{\omega}_o)\cdot (L_{mi}(\vec{x},\vec{\omega}_i)\cdot L_i(\vec{x},\vec{\omega}_i))d\vec{\omega}_i. \quad (1)$$

In Section 4 we will describe how to compute L_{mi} in order to integrate diffuse interreflections as well as in-scattering. In the simplified model described by Equation 1, light is only scattered when traveling towards $\vec{x}$, such that no light is scattered along the viewing direction. Please note, that we use a slightly modified notion for L as compared to L_i and L_{mi}. While L is defined based on the direction ω going out from $\vec{x}$, L_i and L_{mi} are defined based on the direction ω going in to $\vec{x}$.

With the simplified illumination model specified in Equation 1, we are able to incorporate complex material functions specified through M. M, L_{mi} as well as the incident light L_i are defined over the unit sphere Ω. Obviously, the integration over Ω involves a significant computing effort, and cannot be made for each frame. However, when assuming that the functions are continuous, we can exploit SH basis functions in order to approximate Equation 1 and achieve interactive frame rates. In the following subsection we briefly review the main properties of SHs, before describing their integration into our approximation presented in Section 4.

3.1. Spherical Harmonics

Providing a detailed derivation of the concepts behind SHs would be beyond the scope of this paper. Instead, we refer to the main properties exploited by our approach and point out reference to more detailed explanations [Gre03].

SH basis functions allow to represent a continuous function given on the sphere. An infinite number of SH basis functions $Y_l^m(\vartheta,\varphi)$ of varying order l and degree m can be weighted with SH coefficients c_{ij}, to represent arbitrarily complex functions $f(\vartheta,\varphi)$ defined over the sphere:

$$f(\vartheta,\varphi) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} c_{lm}Y_l^m(\vartheta,\varphi).$$

Intuitively, the $Y_l^m(\vartheta,\varphi)$ can be thought of having two independent components to span the sphere. Complex exponential functions with a single argument of period 2π are used to define the variations along the latitude, while associated Lengendre polynomials are used to capture variations along the longitudes. To allow an efficient computation and representation of f, the representation can be bandlimited:

$$\tilde{f}(\vartheta,\varphi) = \sum_{l=0}^{L}\sum_{m=-l}^{l} c_{lm}Y_l^m(\vartheta,\varphi)$$

In this bandlimited version, the number of summands is equal to $1+3+\ldots+(2L+1) = (L+1)^2$, which directly influences the number of required c_{ij}. For practical reasons we have set $L = 3$ throughout this paper, resulting in $(3+1)^2 = 16$ coefficients. To project f and compute the c_{ij}, we use a least square projection, which minimizes the squared error between our approximation and the original function f. Thus, the SH coefficients c_j can be computed as follows:

$$c_j = \int_S Y_j(s)f(s)\,ds. \quad (2)$$

Due to the orthonomality of the SH basis functions, it can be shown, that the integration of two SH-projected functions f and g becomes a sum of the products of their coefficients c_i and d_i:

$$I(\tilde{f},\tilde{g}) = \sum_{i}^{n} c_i d_i. \quad (3)$$

This property makes SH integration very efficient, since when interpreting c and d as vectors, the integration can be easily computed on the GPU by using a dot product. Thus, instead of reconstructing and integrating the original functions, just a dot product is be used. Another important SH property is the rotational invariance. When a rotation is desired, it can be directly applied to the SH coefficients. This is very helpful, when using SH representations for lighting, where the light source may rotate around an object.

4. Realizing Advanced Material Effects

While most SH-based material representations for polygonal data are tailored to support one specific material category, i. e., highly reflective or scattering, in volume rendering the material category may change just by modifying the transfer function. When for instance a lower opacity is assigned through the transfer function, stronger scattering can be expected. Therefore, a specialized SH compression method is needed in order to support different material categories.

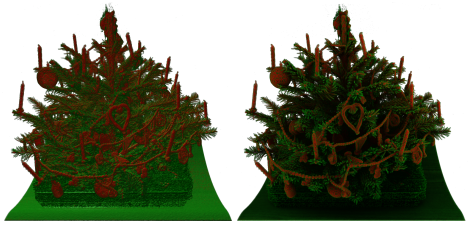
In our approach, the SH projection adapts to the material

function M at each sample position $\vec{x}$. In particular, we consider the transparency $\tau(f(\vec{x}))$ assigned through the transfer function, as well as the gradient length $|\bigtriangledown \tau(f(\vec{x}))|$, depicting the heterogeneity of the medium. To illustrate our approach, we would like to first emphasize the two existing extreme cases. First, when $\tau(f(\vec{x}))$ represents high transparency and $|\bigtriangledown \tau(f(\vec{x}))|$ is small, we can assume that the whole unit sphere centered at $\vec{x}$ affects its illumination. In contrast, when $\tau(f(\vec{x}))$ depicts a rather opaque medium and $|\bigtriangledown \tau(f(\vec{x}))|$ is large, only the front facing hemisphere affects the illumination of $\vec{x}$, since $\vec{x}$ can be considered as a part of a boundary surface [Lev88]. The other possible combinations of the extrema of $\tau(f(\vec{x}))$ and $|\bigtriangledown \tau(f(\vec{x}))|$ lie somewhere between on this scale. So loosely speaking, we consider the whole contribution of the unit sphere, when no surface-like structure can be assumed depending on $\tau(f(\vec{x}))$, and just the hemisphere, when a surface-like structure is present.

For a low-frequency spherical function f, it can be shown that Monte-Carlo integration over a sufficiently high number of samples is sufficient to compute Equation 2 and project f into SH space. The strength of this technique is the ability to generate a uniform distributions of samples over the unit sphere. However, in the case where we deal with a surface-like structure, we need to sample the front facing hemisphere only, since the back facing hemisphere is not assumed to contribute to the illumination of $\vec{x}$. Therefore, we have chosen to integrate an adaptive sampling, in order to perform the SH projection of the environmental lighting. By considering the probability density function, the SH coefficient calculation given in Equation 2 can be reformulated that it equals the expected value E of the random variable r:

$$c_j = \int_S Y_j(s) f(s) ds = \int_\Omega \frac{f(s) Y_j(s)}{p(r)} p(r) ds = E(r).$$

To evaluate this equation, the probability-density function $p(r)$ needs to be known. In our case, it has to fulfill the requirement $\int_\Omega p(r) ds = 1$. Since in the spherical case this area of integration equals 4π, setting $p(r)$ to $\frac{1}{4\pi}$ is a good choice. In our case, we have to incorporate both a hemispherical or spherical region, in order to deal with translucent and surface-like structure properties. Since these properties are continuous, we introduce a below-surface angle α, which is proportional to the transparency of a voxel and lies between 0 and $\frac{\pi}{2}$. During Monte-Carlo sampling, only rays are considered if their angle with $\bigtriangledown \tau(f(\vec{x}))$ is less or equal to $\frac{\pi}{2} + \alpha$. To fulfill the requirements of the probability density, 2π is used as a base value for the hemisphere, and the additional ring below the surface given by α is included as $2\pi + 2\pi \cdot \sin\alpha = 2\pi(\sin\alpha + 1)$. This term is equal to 2π for $\alpha = 0$ and to 4π for $\alpha = \frac{\pi}{2}$, which corresponds to the whole sphere.



(a) (b)

Figure 1: *The Christmas tree data set, with gradient-based lighting (a) and with a color bleeding contribution compressed using SH basis functions (b).*

4.1. Color Bleeding Effects

By using the modified Monte-Carlo integration, the computation of the color bleeding effects can be done easily by performing SH projection. This is a straight forward extension to the diffuse shadowed radiance transfer. However, we omit a weighting with the clamped cosine term, depending on the angle between $\bigtriangledown \tau(f(\vec{x}))$ and the light vector, since a similar behavior is already incorporated in our modified sampling approach described above. Thus, from each voxel, rays are cast in random directions ω_i based on our modified sampling scheme. The color and intensity changes, which occur to the ray while traveling through the medium, are modeled by evaluating the front-to-back compositing volume rendering integral in order to obtain $L_{mi}(\vec{x}, \vec{\omega}_i)$. This is different to the approach by Ritschel, where only absorption and thus no chromaticity effects are considered [Rit07]. After the ray-casting for the current sample has been completed, each of the accumulated color channel values are used as a weight for the values $Y_l^m(\vartheta_s, \varphi_s)$ of the current sample, which are added to the total sum of the SH projection. When substituting the thus computed L_{mi} in Equation 1, renderings as shown in Figure 1 (b) can be generated. When comparing the outcome with conventional gradient-based shading (see Figure 1 (a)), it can be seen that a better depth separation can be achieved.

In this approach, we can use two different ways of incorporating material properties. When evaluating the material function M for each sample on the ray, it directly influences the bleeding color. However, due to the fact that M is considered in the SH projection, it cannot be exchanged without performing a new SH projection. Alternatively, M could be neglected during the ray-marching and instead the transfer function color could be taken into account. In this case, material functions could be exchanged during rendering interactively, as long as they have a similar hue.

4.2. Scattering Effects

With the approach described in the previous subsection, we are able to incorporate indirect illumination effects by ex-

(a) (b) (c) (d)

Figure 2: *Application of the interactive scattering technique to different volume data sets: For the hand data set, we have set R_d to allow a realistic skin appearance (a). For the engine data set R_d has been set to mimic a porcelain-like appearance (b). For the teapot an R_d resulting in more scattering has been used (c), while the bonsai shows the occurrence of the bleeding colors due to the use of brown and green (d).*

ploiting the computation of L_{mi}. However, to further increase the level of realism for certain materials, scattering effects need to be incorporated. Hao and Varshney could show in the context of subsurface scattering, that although scattering is in principle a global effect, its influence is rather local due to the exponential falloff of light [HV04]. We have adopted this insight, and exploit an additional scattering pass. Since we have already computed and SH projected the L_{mi} for all $\vec{x}$ as described in the previous subsection, we can exploit this information when computing the scattering contribution. While previous interactive approaches have constrained scattering just to a cone angle [KPH*03, RDRS10], we are thus able to perform the scattering over the whole unit sphere, or the hemisphere depending on the adapted Monte-Carlo sampling. When assuming, that multiple scattering dominates, the in-scattering reaching $\vec{x}$ from its neighborhood can be computed as follows:

$$L'_{mi}(\vec{x},\vec{\omega}_i) = \sum_{\vec{x}_i \in N(\vec{x})} (\frac{1}{\pi} R_d) \cdot L_{mi}(\vec{x}_i,\vec{\omega}_i) \cdot \frac{1}{|N(\vec{x})|} \qquad (4)$$

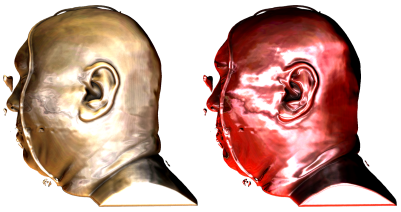
We have specified this equation in its discrete form to depict, that this additional scattering pass is computed based on the already present SH coefficient volumes, containing the coefficients for L_{mi}. Since these volumes are by nature given in a discrete data structure, the $\vec{x}_i$ in Equation 4 refer to actual voxel positions instead of sample positions. Equation 4 is inspired by the subsurface scattering equation introduced by Hao and Varshney [HV04]. Since we are not interested in subsurface scattering, but true volumetric scattering, we have neglected the cosine weighting from the original equation. Furthermore, we have dropped the Fresnel term, and modified the weighting to comply with the volumetric scattering neighborhood $N(\vec{x})$. $\frac{1}{\pi} R_d$ describes the diffuse reflectance obtained by a single dipole approximation for multiple scattering. This approximation achieves a comparable accuracy as using the diffusion approximation with a volumetric source distribution [JMLH01]. Thus, Equation 4 enables us to compute the scattering contribution for each voxel $\vec{x}$ and a given light direction $\vec{\omega}_i$. Notice that the color bleeding contribution is also incorporated in Equation 4, since we use L_{mi} as derived in the previous subsection. The used material parameters R_d have been set as specified by Jensen et al. [JMLH01]. Although, the parameters have been originally acquired to simulate subsurface scattering, we were able to achieve compelling scattering effects by using these parameters.

In our implementation, the neighborhood size $|N(\vec{x})|$ is assumed to be constant within the entire volume. Obviously, the resolution and the average size of structures contained in the data set should be taken into account in order to specify $|N(\vec{x})|$. However, in our case we could always achieve good results, when setting $|N(\vec{x})|$ to 10% of the maximum volume dimension. An alternative approach would be to adapt it based on the material properties of the current region around $\vec{x}$.

The main benefit of this approach is the fact, that we can again use SH projection, in order to efficiently store the L'_{mi} as the results of this scattering pass. Thus, as with conventional SH lighting, the light properties can be changed interactively. Figure 2 shows the application of Equation 1, when we substitute L_{mi} with L'_{mi} and use different parameters for R_d.

4.3. Local Material Effects

Now, that we are able to compute the environmental illumination contribution, i. e., the in-scattering reaching $\vec{x}$, we can focus on the realization of the local material effects. As seen in Equation 1, this is specified by the material function $M(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$. Since M is also a spherical function, we can compress it using standard SH projection techniques. Therefore, we cast rays from the the center of a sphere representing the material to the outside, and project the hit values



(a) (b)

Figure 3: *In this example we use BRDFs as material functions M. The BRDFs have been SH projected by using* 1250 *samples distributed over the hemisphere.*

on the sphere's surface. Since this can be done once in a preprocessing for each material added to the system, performance is not an issue when regarding this precomputation. However, to simplify the rendering process, we have decided to stick with the same number of SH bands as in the other SH projections. At the same time, by increasing the number of samples, we can improve the quality of the approximation. Examples where we have used 1250 samples distributes over the hemisphere, are shown in Figure 3. In this case, we have used data from the MERL BRDF library [MPBM03] to describe the material function M for surface-like structures having a high degree of reflectance. The actual representation in our implementation is described in Section 5.

We have used the same projection technique to represent light probes, in order to integrate natural lighting conditions depicted as L_i. Thus, we have all necessary factors for computing Equation 1 and can compute the illumination at $\vec{x}$ by evaluating Equation 3. However, we have the problem, that while SH integration can be done easily for two functions (see Equation 3), dealing with three functions, i. e., M, L'_{mi} and L_i, becomes more complex. While existing techniques allow to realize the integration of three functions [Sny06], we have decided to use a simpler technique due to performance issues. Therefore, we approximate the integration by using the following equation, which is not physical correct but led to convincing visual results:

$$L(\vec{x},\omega_o) = \int_\Omega M(\vec{x},\vec{\omega}_i,\vec{\omega}_o)L_i(\vec{x},\vec{\omega}_i)d\vec{\omega}_i \cdot \int_\Omega L'_{mi}(\vec{x},\vec{\omega}_i)L_i(\vec{x},\vec{\omega}_i)d\vec{\omega}_i.$$

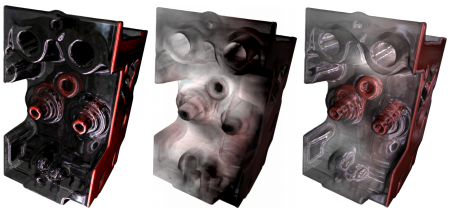
Intuitively, we simplify the triple product to two successive double products. First, we integrate over M and L_i to obtain the local material effects. The result of this integration is modulated by the environmental illumination described by the second integral. Applications of the introduced rendering technique are shown in Figure 4. Figure 4 (a) shows the application of conventional gradient-based shading, while in Figure 4 (b) the color bleeding contribution through L_{mi} has been incorporated and in Figure 4 (c) BRDFs have been assigned to additionally incorporate a complex material function.

5. Implementation

The concepts described in this paper have been integrated into a GPU-based volume ray-caster implemented in C++ by using OpenGL in combination with GLSL. The coefficients for the SH projections of L_{mi} and L'_{mi} are stored in 3D coefficient volumes, whereby those representing L_{mi} can be discarded, after L'_{mi} has been computed, since they are not accessed during rendering. To generate the SH coefficient volumes, we exploit framebuffer objects in conjunction with multiple render targets. The actual ray marching is performed in a fragment shader, while rendering slice by slice into the 3D coefficient texture. While the GLSL specification describes functionality for generating random numbers, this feature is not implemented yet. Therefore, we have decided to generate the random samples for the Monte-Carlo sampling on the CPU in form of normalized cartesian vectors and upload them to the shader as a 2D texture. The same is done for the $Y_l^m(\vartheta,\varphi)$ for each of the sample directions. By using the random samples, rays are cast within the shader. After a ray has been terminated, its contribution is multiplied with the values $Y_l^m(\vartheta_s,\varphi_s)$ obtained from the second 2D texture and added to the total integration sum. Thus, we obtain all $(L+1)^2$ SH-coefficients for the current voxel, which are distributed to the attached render targets.

To speed up the computation, we have integrated early ray-termination based on a ray opacity threshold of 95%. Furthermore, since the ray-casting is performed directly in volume space $[0...1]^3$, a sample ray is terminated if one of its position-coordinates reaches 0 or 1.

Currently multiple render target functionality is limited to 8 simultaneous output targets. Therefore, we have chosen $L = 3$ as the highest band, yielding 16 coefficients per color. We generate these 16 coefficients by rendering into 4 slices with 4 channels simultaneously. However, to reduce memory requirements, we exploit the fact, that human beings are less sensitive to chromaticity variations as compared to varia-



(a) (b) (c)

Figure 4: *Different effects demonstrated to the engine data set: the application of $M \cdot L_i$ (a), the environmental light contribution L_{mi} (b), and the the application of $M \cdot L_{mi} \cdot L_i$ (c).*

tions in luminance [Liv02]. Therefore, we have chosen to use only a 2-band projection for the chromaticity values while keeping the 4-band projection for the luminance. Thus, the three color-coefficient sets sum up to only $3 \cdot 4 = 12$ coefficients altogether, which can be handled using three FBO-attached textures during the projection, as well as three 3D volume textures during rendering. Even when using chromaticity and luminance, this adds up to $4 + 3 = 7$ render targets, which makes this operation possible in a single pass.

While the SH coefficient volumes are generated on the GPU, the projection of the material functions and the light probes is less time critical and can be performed on the CPU, where we generate a partitioned 2D texture representation to be passed to the shader during rendering. When using a 4-band compression, the resulting texture is partitioned as shown in Figure 5. For each ω_o, a combination of texels represents the 16 SH coefficients needed for the ω_i. The texture consists of three columns, having 4 quadrants of size $N \times N$ texels, when using N random samples. The first 4 coefficients (bands 0 and 1) are saved in the lower left, the next 4 in the lower right (the first 4 coefficients of band 2), the next 4 in the upper left (one coefficient of band 2 and 3 coefficients of band 3) and the final 4 in the upper right (the remaining 4 coefficients of band 3). The x-axis of each quadrant represents ϑ of ω_o, ranging from 0 to π, and the y-axis represents φ of ω_o, ranging from 0 to 2π. Thus, we have all needed coefficients in a GPU-friendly format and the integration as described in Subsection 4.3, can be performed during rendering in a fragment shader by using matrix and vector operations.

6. Performance Results

Table 1 shows a comparison of the performance of the SH projection process for different parameters. The tests were performed with three different data sets, having a resolution of $128 \times 128 \times 128$, $256 \times 256 \times 128$ and $256 \times 256 \times 256$ voxel. All tests have been conducted on a Intel Core2 Quad CPU Q9450, running at 2.66 GHz, with 4 GB of main RAM and an nVidia GeForce 9800 GTX with 512 MB RAM. In the columns the SH projection times as well as the frame rates are shown for shadows only, scattered shadows and scattered shadows and color bleeding. During all tests as well as to generate the renderings shown in the paper, we have used downsampled coefficient volumes of about 33% of the original size, and voxels having an opacity of zero have been discarded. Assuming that the radiance transfer is of low-frequency, the downsampling is a simple and effective optimization. Furthermore, we have used increasing ray step size [Rit07]. The windows resolution used to compute the given frame rates was 400×400.

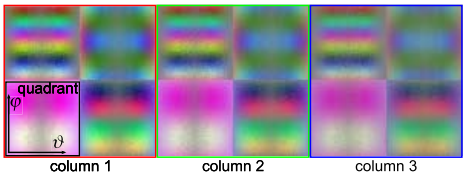


Figure 5: *The BRDF coefficient texture for the aluminum-bronze material. One quadrant holds* 4 *coefficients for each* ω_o*. The x-axis spans the values* $\vartheta = 0...\pi$ *and the y-axis the values* $\varphi = 0...2\pi$*.*

	Shadows	+Scattering	+Bleeding
128^3	0.9s / 24 fps	21.3s / 17 fps	58.9s / 13 fps
$256^2 \times 128$	2.6s / 20 fps	70.2s / 14 fps	180s / 5 fps
256^3	3.4s / 6 fps	149s / 3 fps	388s / 1 fps

Table 1: *Timings for SH projection and rendering with data sets of different voxel sizes. From left to right: shadows only, scattered shadows, scattered shadows with color bleeding.*

As it can be seen in Table 1, for data sets of $128 \times 128 \times 128$ voxels, we achieve interactive frame rates for all compared SH techniques. This allows to inspect the data set for a given transfer function interactively. Furthermore, for the shadowing technique the SH projection is still fast enough to allow to change the transfer function. Obviously, the scattering pass results in an additional performance penalty, such that the transfer function cannot be changed without waiting for the SH projection. However, since the reprojection has to be performed only when the transfer function is changed, the actual rendering can still explored at $14-17$ frames, i. e., camera and light parameters can be changed freely. As described in Subsection 4.1, the material function M can only be exchanged interactively when it is not considered during the SH projection. In fact, in many application areas of medicine or computational fluid dynamics, predefined transfer functions are used, which could also involve M. Table 1 also shows, that when additionally incorporating the color bleeding during the shadowing pass, both SH projection as well as rendering take significantly longer. This delay is due to the fact, that three additional SH coefficient volumes have to be generated and accessed during rendering.

7. Conclusions and Future Work

In this paper we have demonstrated how to integrate realistic light material interaction effects into interactive volume rendering. With the proposed technique we are able to exploit material functions of reasonable frequency. By using a modified SH projection, which has been adapted to the needs of volume rendering, we are able to represent these effects using a common representation. As we have shown, this also allows the seamless integration of conventional SH lighting effects. To our knowledge, this is the first application of non cone-shaped phase functions in the area of interactive volume rendering. To achieve this, we have explained how to incorporate the desired effects within the volume rendering

integral, and have described our hybrid CPU/GPU implementation. We believe that the proposed concepts bring the realism of volume rendered images to a new level.

While we were able to produce high-quality imagery at interactive frame rates, there are still several open issues, which could potentially be addressed in the future. The main drawback of SH-based techniques is the restriction to approximate low-frequency functions only. While this is sufficient for most phase functions, for some materials having sharp specular highlights this might not be sufficient and Gibbs ringing may occur. Therefore, alternative concepts should be investigated for these scenarios. Such as the SH approximation method by Zafar et al. [ZARM06], or wavelet compression techniques, which have been proven useful in the area of polygonal rendering [SM06], and might be also used in DVR. Another issue with the proposed technique is the availability of appropriate material functions. While several simplified models exist, having a true volumetric capturing would be of great interest. Finally, since the presented technique can be considered as approximative, it should be evaluated with respect to perceptual requirements.

Acknowledgments

This work was partly supported by grants from Deutsche Forschungsgemeinschaft, SFB 656 MoBil (project Z1). We would like to thank the reviewers for their constructive comments. The presented concepts were implemented using the Voreen volume rendering engine (http://www.voreen.org).

References

[BB09] BANKS D. C., BEASON K.: Decoupling illumination from isosurface generation using 4d light transport. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1595–1602. 2

[BG07] BRUCKNER S., GRÖLLER M. E.: Style transfer functions for illustrative volume rendering. *Computer Graphics Forum 26*, 3 (2007), 715–724. 2

[BGB*06] BEASON K. M., GRANT J., BANKS D. C., FUTCH B., HUSSAINI M. Y.: Pre-computed illumination for isosurfaces. In *Visualization and Data Analysis '06 (SPIE Vol. 6060)* (2006), pp. 1–11. 2

[Gre03] GREEN R.: Spherical harmonic lighting: The gritty details. In *Game Deveopers Conference* (2003). 3

[HKSB06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K.: GPU-accelerated deep shadow maps for direct volume rendering. In *ACM SIGGRAPH/EG Conference on Graphics Hardware (GH)* (2006), pp. 49–52. 1

[HV04] HAO X., VARSHNEY A.: Real-time rendering of translucent meshes. *ACM Trans. Graph. 23*, 2 (2004), 120–142. 5

[JMLH01] JENSEN H., MARSCHNER S., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *ACM SIGGRAPH* (2001), pp. 511–518. 5

[KH84] KAJIYA J. T., HERZEN B. P. V.: Ray tracing volume densities. In *ACM SIGGRAPH* (1984), pp. 165–174. 2

[KPH*03] KNISS J., PREMOZE S., HANSEN C., SHIRLEY P., MCPHERSON A.: A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (2003), 150–162. 1, 2, 5

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3 (1988), 29–37. 4

[LHY09] LJUNG P., HERNELL F., YNNERMAN A.: Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 15*, 2 (2009). 1, 2

[Liv02] LIVINGSTON M.: *Vision and Art: The Biology of Seeing*. Harry N. Abrams, New York, 2002. 7

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108. 2

[MPBM03] MATUSIK W., PFISTER H., BRAND M., MCMILLAN L.: A data-driven reflectance model. *ACM Transactions on Graphics 22*, 3 (July 2003), 759–769. 6

[PBVG10] PATEL D., BRUCKNER S., VIOLA I., GROELLER E.: Seismic volume visualization for horizon extraction. In *IEEE Pacific Visualization (PacificVis 2010)* (2010). 2

[RDRS10] ROPINSKI T., DÖRING C., REZK-SALAMA C.: Interactive volumetric lighting simulating scattering and shadowing. In *IEEE Pacific Visualization (PacificVis 2010)* (2010). 2, 5

[Rit07] RITSCHEL T.: Fast GPU-based Visibility Computation for Natural Illumination of Volume Data Sets. In *Short Paper Proceedings of Eurographics 2007* (2007), pp. 17–20. 2, 4, 7

[RMSD*08] ROPINSKI T., MEYER-SPRADOW J., DIEPENBROCK S., MENSMANN J., HINRICHS K. H.: Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum (Eurographics 2008) 27*, 2 (2008), 567–576. 1, 2

[Sal07] SALAMA C. R.: GPU-based monte-carlo volume raycasting. In *Pacific Graphics (PG)* (2007). 1, 2

[SKS02] SLOAN P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM SIGGRAPH* (2002), pp. 527–536. 2

[SM06] SUN W., MUKHERJEE A.: Generalized wavelet product integral for rendering dynamic glossy objects. In *ACM SIGGRAPH* (2006), pp. 955–966. 8

[Sny06] SNYDER J.: *Code generation and factoring for fast evaluation of low-order spherical harmonic products and squares*. Tech. rep., Microsoft Corporation, 2006. 6

[SPH*09] SCHOTT M., PEGORARO V., HANSEN C., BOULANGER K., BOUATOUCH K.: A directional occlusion shading model for interactive direct volume rendering. In *Computer Graphics Forum (Eurographics/IEEE VGTC Symposium on Visualization 2009 issue)* (2009), pp. 855–862. 1, 2

[WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. *ACM SIGGRAPH 26*, 2 (1992), 255–264. 2

[WPHS06] WYMAN C., PARKER S., HANSEN C., SHIRLEY P.: Interactive display of isosurfaces with global illumination. *IEEE Transactions on Visualization and Computer Graphics 12*, 2 (2006), 186–196. 2

[ZARM06] ZAFAR N. B., AKESSON J., ROBLE D., MUSETH K.: Scattered spherical harmonic approximation for accelerated volume rendering. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches* (2006), p. 148. 8