In IEEE Transactions on Visualization and Computer Graphics 2011.

Extinction-based Shading and Illumination in GPU Volume Ray-Casting

Philipp Schlegel, Maxim Makhinya, and Renato Pajarola, *Member, IEEE*

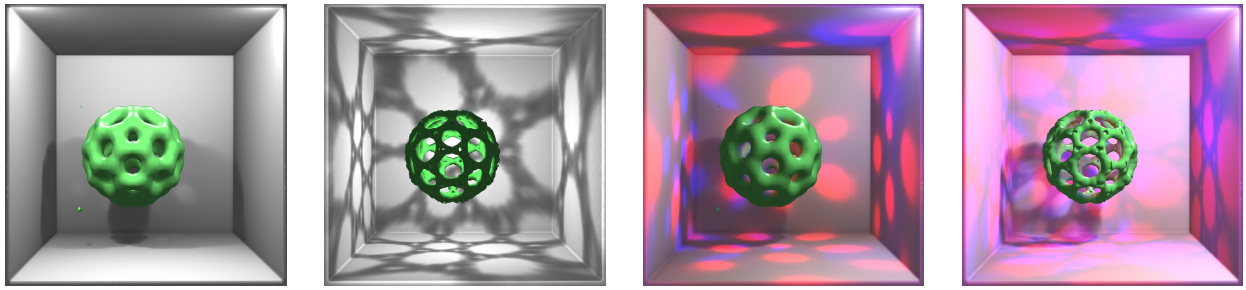

Fig. 1. Bucky ball in a box illuminated by up to three point lights inside the volume (two white point lights in top front box corners; one white light inside bucky ball; two examples of one white light in top-right box corner and two colored lights inside bucky ball).

Abstract—Direct volume rendering has become a popular method for visualizing volumetric datasets. Even though computers are continually getting faster, it remains a challenge to incorporate sophisticated illumination models into direct volume rendering while maintaining interactive frame rates. In this paper, we present a novel approach for advanced illumination in direct volume rendering based on GPU ray-casting. Our approach features directional soft shadows taking scattering into account, ambient occlusion and color bleeding effects while achieving very competitive frame rates. In particular, multiple dynamic lights and interactive transfer function changes are fully supported.
Commonly, direct volume rendering is based on a very simplified discrete version of the original volume rendering integral, including the development of the original exponential extinction into α-blending. In contrast to α-blending forming a product when sampling along a ray, the original exponential extinction coefficient is an integral and its discretization a Riemann sum. The fact that it is a sum can cleverly be exploited to implement volume lighting effects, i.e. soft directional shadows, ambient occlusion and color bleeding. We will show how this can be achieved and how it can be implemented on the GPU.

Index Terms—Volume Rendering, Shadows, Ambient Occlusion, GPU Ray-Casting, Exponential Extinction

1 Introduction

Direct volume rendering (DVR) [6] is one of the most useful and popular methods for visualizing volumetric scalar field datasets without explicitly extracting geometry. Typically the visualization of such volume datasets is a critical tool in bio-medical imaging applications for CT, MRI or PET scan data. It is also used for the analysis of numerical simulations and occurs in visual arts. Visualizing sampled volume data involves the reconstruction of the continuous scalar field from the discrete dataset followed by the evaluation of the direct volume rendering integral [19, 22]. Even though many algorithms have been proposed to implement this, like splatting [34] and texture slicing [7], recently GPU-based ray-casting [33] has become the preferred choice for many scientific visualization applications due to its image quality, performance and conceptual simplicity.

Advanced volume illumination models such as directional soft shadows, ambient occlusion or color bleeding effects can further enhance the visual perception and spatial impression of GPU ray-casting. A lot of research has been done on such advanced illumination models. Many of these are standard in classical surface ray-tracing but not in scientific volume visualization applications. The problems with adapting such models to GPU ray-casting for DVR are numerous. Direct volume rendering on the GPU should be interactive and very responsive to the user while exploring the dataset. Hence, the available processing resources per rendered frame for evaluating an advanced illumination model are quite limited. Furthermore, when dealing with volumetric datasets, no explicit geometry is given and the material properties are defined by an arbitrarily shaped transfer function (TF) that can change at run-time. This typically prohibits an expensive preprocessing step relying on a static TF or static light positions. Generally, expensive preprocessing is undesirable because in many domains it is not feasible to undergo a time-consuming preprocess before data exploration, especially for time-varying datasets. Lastly, the advanced illumination models must be implemented on the GPU, fitting its computational model and its limited on-board memory resources.

The qualifying idea of our novel approach to address these problems is a solution based on the original exponential extinction coefficient of the DVR integral, and an efficient GPU-based real-time summation mechanism. Besides the more accurate evaluation of the volume rendering integral, the beneficial extinction coefficient property of being additive allows discrete summation order independently when sampling along a ray. Throughout this paper special algorithms and data structures will be elaborated based on the original exponential extinction coefficient in order to solve the above outlined lighting problems and to get fast, plausible volume illumination and shading effects.

1.1 Related Work

Although efficient, simple Phong-Blinn volume shading does not provide enough realism and has poor global depth reproduction. While computationally inexpensive techniques, such as semitransparent halos [2] or directional occlusion [31] can easily be implemented on

• *Philipp Schlegel, Maxim Makhinya, and Renato Pajarola are with Visualization and MultiMedia Lab, Department of Informatics, University of Zürich, Switzerland, E-mail: {schlegel, makhinya}@ifi.uzh.ch, and pajarola@acm.org.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

GPUs to improve depth perception, more advanced lighting models are still a challenge. Soft shadows, ambient occlusion, light scattering and color bleeding are desired features in GPU volume rendering [9].

Soft directional shadows that are able to incorporate infinite light sources only, requiring an additional shadow volume, which is updated on each light position change, were proposed by Behrens and Ratering [1]. Classical shadow mapping [35] can also be applied to volume rendering in limited scenarios to obtain hard shadows. To improve performance for semitransparent objects, deep-shadow maps [26] use multiple opacity layers. Shadow maps are dependent on light position and TF with the evaluation typically done for a single light only. Half-angle slicing [13, 14, 36] is able to produce more realistic soft shadows with very small additional storage requirements, but also with the limitation of a single directional light source support.

Obscurance and ambient occlusion (AO) methods [21, 17] provide a simple way to approximate indirect global illumination by sampling a limited neighborhood. Screen-space AO (SSAO) is popular in polygonal shading [32] as well as in volume rendering [4] due to its simplicity and high performance. These methods rely on the visible pixels' depth value estimates, are fast and have sufficient quality for opaque objects and simple geometry. SSAO can, however, fail in complex scenarios due to the limited 2D depth information available; they are also inefficient in the case of semitransparent objects when only one depth value per pixel can be used. These limitations can, at an increased cost, partially be solved through depth-peeling [8], where a limited number of depth samples can be used in practice.

Object space AO provides better quality than SSAO, at the price of extra storage and computation. Typically, an additional 3D texture for density values is used, which has to be updated upon TF changes. Depending on the implementation, TF updates can introduce significant lag, if individual neighboring voxels are sampled as presented by Ruiz et al. [28], or it can be fast if aggregate values are considered as shown by Diaz et al. [4]. Similar techniques [18, 23] are suitable for polygonal rendering; additional 3D textures that represent voxelized objects have to be created, updated when geometry changes, and sampled during rendering. Another approach to dynamic AO, based on per-voxel local histograms, was introduced by Ropinski et al. [27]. TF and light source independent illumination are achieved by convolving local histograms with the current TF during rendering to obtain an environmental color of each voxel. Additional space for histogram clusters is moderate, and rendering itself is efficient. However, the preprocessing step requires hours even for medium size models.

Scattering of light requires the inclusion of all indirect light contributions. Thus true evaluation is very expensive, yet a clever approximation can provide pleasant visual quality at interactive frame rates on modern hardware. Good results were achieved by Kniss et al. [14], where only light contribution within a cone directed towards the light source was taken into account at each sample position. This allowed superimposing contributions of direct and indirect light in one rendering pass, using half-angle slicing. No preprocessing is required, but only a single point light source can be used. This method was further explored by Ropinski et al. [25], resulting in fewer sampling operations for the illumination computation and allowing integration with GPU volume ray-casters. These benefits are achieved at the expense of recomputing a 3D light volume upon every light position change.

Color bleeding is often integrated with obscurance methods, as it requires only minor changes to AO [20]. Many AO and translucency methods gather aggregate color information of the voxels' neighborhood together with the opacity, to use it for color bleeding effects [28, 27, 11]. Similarly, color bleeding can be integrated in light scattering techniques as well [14, 25].

1.2 Contributions

Our approach includes a number of contributions and advantages to achieve more realistic and efficient volume illumination. (i) A unified approximate model for both local ambient occlusion and directional soft shadows, as well as (homogeneous) light scattering and color bleeding is presented. (ii) The model is based on the summation of the exponential extinction coefficient, exploiting a 3D summed area table (SAT) for fast computation. (iii) Directional soft shadows can be computed reusing the same SAT as for ambient occlusion, having minimal performance impact. (iv) The proposed discrete extinction coefficient summation supports distance weighted ambient occlusion and shadow contributions. (v) The introduced solution requires no expensive preprocessing and allows for interactive TF changes. (vi) Multiple as well as dynamic point and spot light sources are supported. Moreover, arbitrary light source positions are allowed also inside the actual volume data. (vii) The model is integrated in a high-quality ray-casting based volume visualization application and works in real-time on the GPU. (viii) At run-time, some additional 3D texture storage is required to hold the TF dependent extinction coefficients. This data, however, can be adjusted flexibly for resolution accuracy and storage overhead.

2 Lighting with Additive Exponential Extinction

2.1 Additive Exponential Extinction Coefficients

Direct volume rendering is based on the emission and absorption theorem by Max [19], leading to the DVR integral (Eq. 1). It can be shown [22] that the volume rendering integral cannot be solved analytically without making some confining assumptions, and consequently needs to be approximated. This includes the commonly used development of the original extinction coefficient into a Taylor series where only the first two elements are considered, which is equivalent to classical α-blending [24]. However, this is a rather coarse approximation optimized for mapping to fixed-function 3D graphics hardware of the past. Today with fast, programmable GPUs it is not required anymore [15, 30]. A closer approximation based on the original extinction coefficient can be chosen for a more accurate evaluation of the volume rendering integral. In addition, the advantage of integrating over the exponential extinction coefficient is that it corresponds to a summation, since being additive, when sampling along a ray, in contrast to the (ordered) product of α-blending.

In Eq. 1, a ray from $s = 0$ at the back of the volume to $s = D$ at the eye position is considered. The extinction coefficient is indicated by $\tau(s)$, and $E(s)$ is the light reflected or emitted by a volume sample at s. The integrated intensity along a viewing ray is now given by:

$$I(D) = \int_0^D E(s)\tau(s)e^{-\int_s^D \tau(t)dt}ds. \tag{1}$$

In the discretization of Eq. 1 using a step size Δt along the ray, instead of performing a Taylor series expansion and simplification of the extinction term – as done in the past for fixed-function graphics hardware α-blending – the original exponential extinction coefficient can be retained as

$$I(D) \approx \sum_{i=0}^{D/\Delta t} E_i \cdot \Delta t\,\tau_i e^{-\sum_{j=i}^{D/\Delta t}\Delta t\,\tau_j}, \tag{2}$$

where the reflected and emitted light E_i is typically replaced by a voxel color modulated by a simple lighting model.

The formulation of Eq. 2 is sufficiently simple and can easily be implemented on programmable GPUs. The additive property of τ allows for the summation of the samples in a shader in arbitrary order, followed by an exponential function applied to this sum, which can be done efficiently on today's GPUs. Hence the big advantage of Eq. 2 is not only an improved image quality compared to the less accurate approximation using multiplicative α-blending (see also [15, 30]), but also the fact that any attenuation calculation can be implemented by a summation of extinction coefficients, notably in arbitrary order.

In the following we will show how this summation can efficiently be exploited for volume shading and illumination purposes. The basic premise is that any light occlusion and thus shadowing effects arise from the attenuation of light traveling or being scattered through the volume along a ray or within some specific region. Therefore, any light attenuation stems from some extinction factor $e^{-\sum_j \Delta t\,\tau_j}$ where the sum $\sum_j \Delta t\,\tau_j$ must be taken over a ray or region of the volume.

2.2 Ambient Occlusion and Color Bleeding

Ambient occlusion (AO) is an approximated attenuation of diffusely reflected ambient light through occlusion. It is not physically accurate, but since full-fledged physical illumination models such as global illumination [12] are beyond interactive volume rendering, AO is a very useful and effective approximation of the effect [17, 27, 11, 4].

The reflected light term E in the DVR integral includes an ambient term representing diffusely reflected ambient light. In the AO lighting model, this term I_{AO} is not a constant but a function taking the occlusion in the local neighborhood of a sample s into account. It represents the total amount of unoccluded incident light over a sphere Ω at s, where $I(s,\omega')$ denotes the incoming light at position s from direction ω':

$$I_{AO}(s) = \int_{\omega' \in \Omega} I(s,\omega') d\omega'. \tag{3}$$

Instead of densely sampling the sphere Ω and tracing many shadow rays for $I(s,\omega')$, as shown by Ruiz [28] and Hernell [11] respectively, we opt for a much faster approximation. Only a well-defined local neighborhood $\mathbf{N}(s)$ of s is considered for *local ambient occlusion*. We assume that for all samples s a constant ambient light intensity I_A is incident over the boundary $\partial\mathbf{N}(s)$. I_A is proportional to the sum of all lights, expressed by an ambient light term coefficient. Hence only the local light attenuation inside the neighborhood $\mathbf{N}(s)$ has to be considered. The local ambient occlusion is thus modeled by the distribution of the extinction τ in the neighborhood $\mathbf{N}(s)$ as

$$I_{AO}(s) \approx I_A \cdot e^{-\int_{t \in \mathbf{N}(s)} \frac{\tau(t)}{|s-t|^2} dt}, \tag{4}$$

where the inverse of the square distance accounts for the law of radial distance based light attenuation.

Color bleeding describes the phenomenon that the color appearance of a surface is locally affected by colored nearby objects [20]. As this illumination effect is also primarily based on the local neighborhood of a sample point, it can be approximated in a similar way to ambient occlusion. For AO only the extinction coefficient has been taken into account as an indicator for opacity at a particular position within the volume. For the estimation of color bleeding, the color also has to be taken into account as an additional parameter C_{RGB} depending on the TF. So Eq. 4 can be reformulated to

$$I_{AO_{RGB}}(s) \approx I_A \cdot e^{-\int_{t \in \mathbf{N}(s)} \frac{\tau(t) C_{RGB}(t)}{|s-t|^2} dt}, \tag{5}$$

where $I_{AO_{RGB}}$ is a vector describing the intensity per color.

Since the summation of the exponential extinction coefficients in Eqs. 4 and 5 is order independent, it is possible to exploit highly efficient aggregate summation methods (aggregate query q), as described in Section 3. This yields results similar in quality at much higher speeds as demonstrated in Fig. 2.

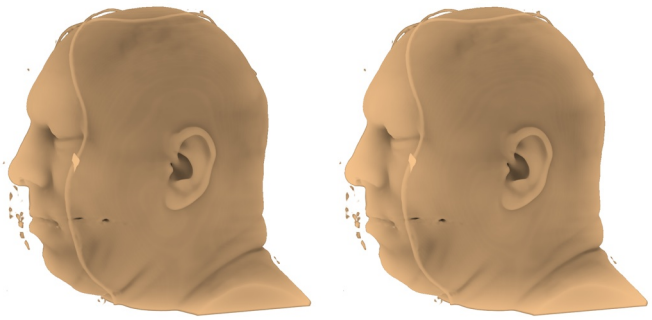

Fig. 2. Ambient occlusion with the neighborhood individually sampled (left) and aggregate extinction coefficients sampling (right). There is virtually no difference regarding image quality, but aggregate sampling is over 45 times faster (0.22s vs 10s).

2.3 Directional Soft Shadows and Scattering

In the context of direct volume rendering, typically semitransparent surfaces and structures are displayed, partially obstructing other surfaces and structures. Thus for evaluating the volume illumination model the question is not whether a light source is obstructed or not but to what degree the light from this source is absorbed while traveling through the volume to the sample point in question. When using an extinction-based model as described in Section 2.1, this can be achieved by casting shadow rays to the light source, densely sampling along these shadow rays and summing up the weighted extinction coefficients. Applying the exponential function to this sum then results in a factor being a measure of how much light from this light source is attenuated before reaching the sample point.

One can imagine that densely sampling many shadow rays for each light source from each volume sample is already very expensive, multiplying the basic volume ray-casting costs by a large factor. Moreover, light scattering describes a process where non-uniformities in the (semitransparent) medium force the traversal of light to deviate from the straight trajectory, caused by reflection of tiny particles. The ratio of light hitting a particle and the reflection direction are random but can be approximated by the *Bidirectional Scattering Surface Reflectance Distribution Function* $R(s,\omega,\omega')$, where s is the scatter position, ω the direction of reflection (to the viewer) and ω' the direction of incoming light as illustrated in Fig. 3. Max [19] accounted for this by adding a scattering term to the volume rendering integral Eq. 1,

$$I(D) = \int_0^D (E(s) + S(s,\omega))\tau(s) e^{-\int_s^D \tau(t) dt} ds, \tag{6}$$

where $S(s,\omega) = R(s,\omega,\omega')I(s,\omega')$, and $I(s,\omega')$ is the incoming light reaching s from direction ω'. When scattering occurs in multiple directions as it is the case in high albedo media, all directions ω' have to be considered by integrating the scattering over the unit sphere.

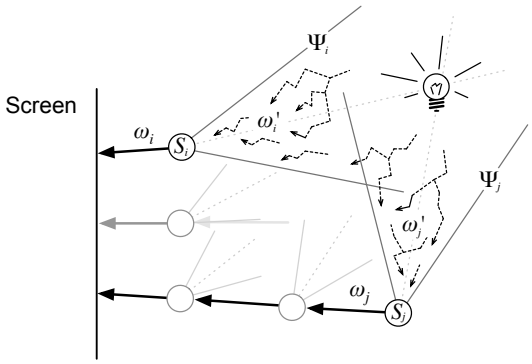

Fig. 3. Approximating scattering effects by considering cones Ψ instead of rays.

According to Max [19] it is overkill to compute multiple scattering for most scientific visualizations, apart from being an elusive goal for interactive rendering. Instead of densely sampling shadow rays or computing multiple scattering, we suggest a very fast solution that produces similar effects with sufficient quality for most applications. The basic idea is not to cast a shadow ray from a sample to the light but a cone [14]. Sampling this cone not only yields the necessary extinction of the light on its way to the sample but also estimates the amount of light scattered towards the sample as a function of the distance and the neighborhood of the ray. A more sparsely occluded neighborhood is an indicator that there is more light scattered to the sample, thus supporting visually plausible soft shadow borders as presented in Fig. 4.

The attractive feature of our method is that the shadow cone does not need to be sampled. To incorporate an estimate of the scattering term S into the discretized volume rendering integral of Eq. 2, we evaluate shadow and scattering effects together by aggregating the extinction values within the shadow cone. Consequently scattering is not a separate term, but is included in E_i.

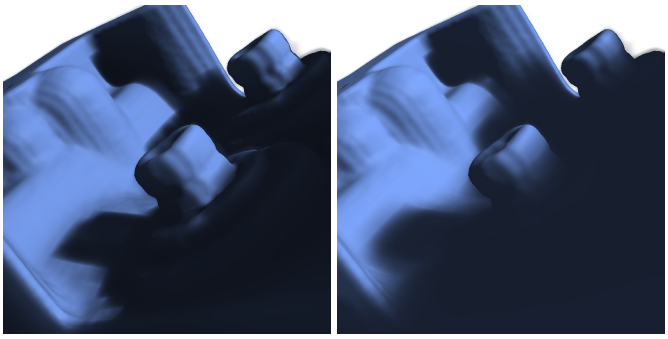

Fig. 4. The engine with classical shadow rays (left) and our soft shadows (right). Even though an entire cone is considered for soft shadows compared to a single shadow ray, the shadow computation is 40% faster (0.0079s vs. 0.0132s).

Commonly, the amount of light E_i reflected or emitted by a voxel is modeled as the sum of an ambient I_A, a diffuse I_D and a specular term I_S, where the ambient term is replaced by our AO term I_{AO} introduced in the previous section. The diffuse and specular terms consist of the incoming light intensity I_L from all light sources, multiplied by material properties of the voxel (i.e. color C_{RGB}) as well as angle-dependent diffuse and specular reflection factors respectively. To incorporate shadows, the light intensity I_L is further attenuated by a factor τ_L due to any occluders between the light source(s) L and voxel s: $\tau_L = e^{-\int_s^L \tau(t)dt}$. However, in our approach the integral is not only evaluated along a single shadow ray but over a cone Ψ towards the light source (see also Fig. 3): $\tau'_L = e^{-\int_{t\in\Psi} h(t)\tau(t)dt}$ where h is a weighting function. Hence, the inclusion of scattering is the extension of τ_L to τ'_L as an attenuation factor of I_L, replacing the specific scattering term S, giving rise to a shadowed and scattered lighting term

$$I_{Sh}(s) \approx I_L \cdot e^{-\int_{t\in\Psi_s} h(t)\tau(t)dt}. \tag{7}$$

The key feature of our implementation is that the summation $\int_{t\in\Psi} h(t)\tau(t)dt$ is approximated by a series of aggregate extinction queries as described in Section 3. The weighting function $h(t)$ is the inverse of the aggregate query size V_q^{-1}. Due to the query size growing with the distance and the cone diameter, the influence of occlusion and scattering in these areas decreases rapidly.

The limitation of our approximation to scattering is the assumption that the scattering function R is constant for all directions and that the amount of scattering is basically proportional to the extinction coefficient of the medium. The rationale behind this is that the amount of light being absorbed or reflected directionally by the medium cannot be scattered isotropically or in a forward manner. Typically more light is absorbed or reflected directionally by denser media, especially when comparing gases to solids. In order to model medium specific scattering properties a separate transfer function would have to be applied. However, adjusting the cone angle allows for a certain flexibility. A narrow cone approximates forward scattering, taking a limited range strongly into account whereas a wider cone approximates more isotropic scattering, taking a broad range into account but with far less influence of individual voxels. Scattering in relation with cones is also discussed extensively by Kniss et al. [14]. Generally, our approach works very well for typical applications of scattering in highly homogeneous media such as smoke or a block of (wax-like) translucent material (Fig. 9).

3 Implementation

The basis for our implementation is a GPU volume ray-caster [16, 29] built using OpenGL and GLSL shaders. Algorithm 1 shows an overview of the rendering pipeline. Basically a so-called light cache (texture T) is computed containing the summation terms for AO/color bleeding and the directional shadows in the different channels of the texture. During the ray-casting pass, these terms are fetched from the texture with a single lookup and used in the adapted illumination computation. In the beginning it has to be determined if either the TF or the light position relative to the dataset has changed. If this is not the case, the image can be rendered immediately, fetching the information for AO/color bleeding and the directional shadows from the light cache texture. If the TF has changed, the light cache texture needs to be recomputed. For this, a 3D summed area table (SAT, texture S) [3] is constructed in a first step and then the light cache texture is computed in a second step. If only the light position has changed relative to the dataset, it is sufficient to recompute only the part of the light cache texture containing the values for the directional shadows.

Algorithm 1 Overview of the rendering pipeline

```
1:  for each frame do
2:      if transfer function changed then
3:          Apply transfer function to volume V and store result to texture T
4:          /* ———————— SAT ———————— */
5:          Compute SAT from texture T by
6:          - Recursive doubling
7:          - Using ping-pong textures S, T
8:          Store SAT to S
9:          /* ———— Ambient/color bleeding factors ———— */
10:         for each voxel of S do
11:             Sample shells from S
12:             Sum up weighted shells
13:             Store sum to texture T
14:     if light source or transfer function changed then
15:         /* ———— Directional shadow factors ————- */
16:         for each light source L do
17:             for each voxel of S do
18:                 Query cuboids towards light source L
19:                 Sum up weighted cuboids
20:                 Store sum to texture T
21:     /* ———————— Ray casting ———————— */
22:     for each pixel on screen do
23:         Compute entry and exit point for volume V
24:         Compute ray R from entry and exit point
25:         for each sample position P along ray R do
26:             Lookup volume V and apply transfer function
27:             Lookup texture T
28:             Evaluate illumination model
29:             Add contribution to pixel
```

3.1 3D Summed Area Table for Illumination

For any illumination computation I_{AO} or I_{Sh}, as outlined in the previous section, we need to account for an attenuation factor $\tau_L = e^{-\int_\Omega \tau(t)dt}$ that will be multiplied with the light source intensity for ambient occlusion or directional soft shadows. In a discretized setup, this amounts to the computation of the sum of extinction coefficients $\sum_\Omega \Delta t\,\tau_j$, where the additive aggregation of extinction values τ_j is done over a voxel neighborhood $\Omega_A = \mathbf{N}(s)$ for ambient occlusion (and the ambient light I_A is modulated), along a ray $\Omega_L = \text{line}_{\text{voxel } s \text{ to light source}}$ for hard shadows and within a cone $\Omega_L = \Psi$ for soft shadows (and the light source intensity I_L is modulated). Taking ambient occlusion or shadowing into account, the reflected light E_i in Eq. 2 of a voxel due to a light source is basically

$$E_i = I_{A,L} \cdot e^{-w_{A,L}\sum_{\Omega_{A,L}} \Delta t\,\tau_j} \cdot k \cdot C_{RGB}, \tag{8}$$

where the weighting function $w_A = r_q^{-2}$ is the inverse square of the radius of the aggregate query, and $w_L = V_q^{-1}$ is the inverse of the aggregate query size, and k simply represents a normal, gradient dependent local illumination model factor.

For the fast extinction summation over $\Omega_{A,L}$, instead of using the traditional expensive shadow ray generation and sampling approach, we implement this aggregation operation using a summed area table (SAT) [3] of the extinction coefficients. With a 3D SAT it is possible to derive the sum of all elements inside an arbitrary cuboid in constant time using at most eight table lookups. As shown below, we approximate the extinction summations over $\Omega_{A,L}$ by cuboid SAT queries.

Since the extinction coefficients are transfer function (TF) dependent, this SAT needs to be updated whenever the TF changes. However, fast SAT construction on the GPU [10, 4] can be implemented based on the recursive doubling technique [5] using a logarithmic number of passes, allowing interactive TF changes as demonstrated in Section 4. We use a render-to-3D texture approach which allows for a number of implementation synergies and avoids OpenGL-CUDA switches. Unlike Diaz et al. [4] we are not using opacity values for the SAT but extinction coefficients.

In order to compute our illumination model two auxiliary 3D textures are used, one for the SAT, and another as a ping-pong texture during SAT generation becoming a light cache during rendering. These two textures can be of arbitrary size within the OpenGL limitations, depending on the desired quality/performance, and do not necessarily need to match the input volume resolution, see also Fig. 11. Algorithm 1 shows an overview of the rendering steps.

3.2 Ambient Occlusion and Color Bleeding

Remarkably, for approximating $I_{AO}(s)$ according to Eq. 4, the extinction coefficient SAT can be effectively used. The discretized extinction coefficient summation $\sum_{\Omega_A} \Delta t\, \tau_j$ in Eq. 8 is approximated by a series of cuboid shells as indicated in Fig. 5, where the number and size of the shells can be varied. Hence, Ω_A is a set of cuboids Sh_i. For each shell, its aggregate sum of extinction coefficients can be obtained quickly by SAT lookups. A larger set of shells with varying diameters leads to a better image quality but requires more SAT lookups increasing the costs. According to our experiments as few as three shells are sufficient to reach an image quality hardly distinguishable from individually sampling a large neighborhood, as demonstrated in Fig. 2. Only if the radius of Ω_A exceeds 10% of the radius of the entire dataset, more shells may become necessary. The use of cuboid shells is entirely different from Diaz' approach [4], where the neighborhood is subdivided into eight adjacent octants preventing a distance based weighting.

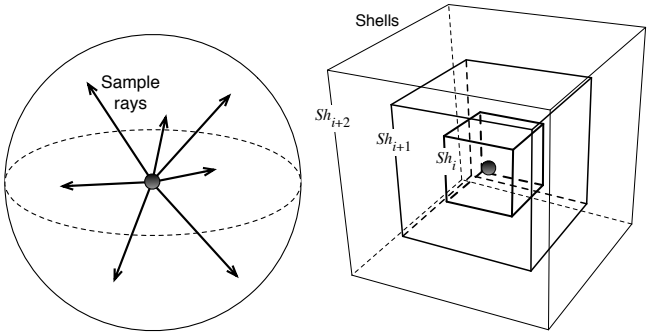

Fig. 5. Ambient occlusion computation by way of sampling the spherical neighborhood (left) versus SAT-based lookups (right). Compared to per-voxel sampling, the number of 3D texture fetches is a order of magnitude smaller using the SAT method.

For AO/color bleeding, multiple shells are queried and accumulated as indicated in Fig. 5. First, the innermost shell Sh_0 is queried from the SAT and weighted by the inverse square of its radius, $\tau_{Sh_0} = SAT(Sh_0) \cdot \left| r_{Sh_0} \right|^{-2}$. Iteratively all shells are accumulated by $\tau_{Sh_{i+1}} = \tau_{Sh_i} + (SAT(Sh_{i+1}) - SAT(Sh_i)) \cdot \left| r_{Sh_{i+1}} \right|^{-2}$ until the last shell is processed. The result of this summation is stored in the auxiliary 3D light cache texture.

Ambient occlusion is independent of the light position, but needs to be recomputed if the TF changes. The actual values for AO are cached together with the values from the directional shadows in the 3D light cache texture. Consequently ambient occlusion in our solution comes at zero cost during rendering.

Of course Eq. 5 for color bleeding can be computed similar to Eq. 4 using a SAT that stores vectors τC_{RGB}. Since four values can be processed per operation with OpenGL textures, the SAT for τC_{RGB} can be constructed at the same time with the SAT for τ, and stored in the same 3D texture at no additional computation costs. The only downsides are the additional memory and memory bandwidth requirements compared to a single channel texture that would be used when constructing the SAT for τ only. However, on our hardware the additional memory bandwidth requirements do not harm rendering performance. The typical number of shells required for color bleeding proved to be the same as for AO. An example of color bleeding is shown in Fig. 6.

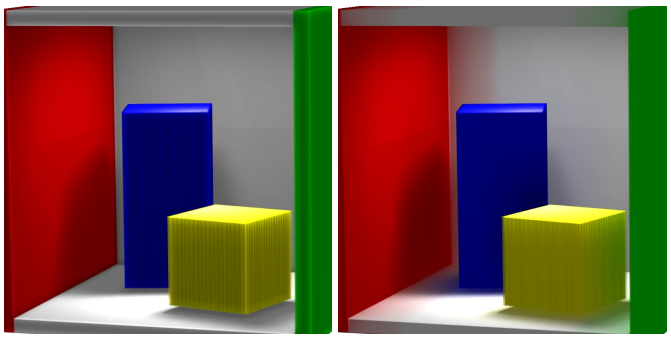

Fig. 6. The Cornell box with soft shadows and strong ambient occlusion (left), as well as color bleeding (right). Due to the fixed light source of the Cornell box, rendering with soft shadows and AO/color bleeding enabled comes at near zero extra cost (one additional texture lookup).

3.3 Directional Soft Shadows and Scattering

For the directional soft shadow illumination $I_{Sh}(s)$ according to Eq. 7, two cases have to be differentiated. If the light sources are at a fixed position with respect to the dataset, as it is the case with the Cornell box model, the attenuation factors for directional soft shadows only have to be computed once and are stored in the auxiliary light cache together with the terms for ambient occlusion/color bleeding. In this case, the total cost for evaluating our extinction-based illumination model during rendering consists of a single, additional texture lookup per sample having only a minor impact on the overall performance. If the light sources change their relative position with respect to the dataset when rotating, moving and zooming, then the occlusion factors have to be queried from the extinction SAT for every frame.

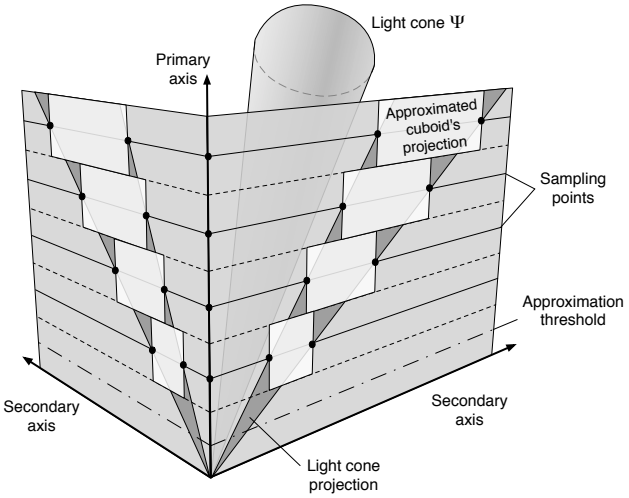

Fig. 7. The cone is approximated by a series of cuboids. The main axis is determined and the cone is projected onto the planes with the secondary axes. The intersections of the projections with lines parallel to the secondary axes through the sample points on the main axis define the cuboids.

When needed, the attenuation factors for directional soft shadows, given by the discrete extinction coefficient summation $\sum_{\Omega_L} \Delta t\, \tau_j$ in Eq. 8 over the sampling cone $\Omega_L = \Psi$, are computed by a render-to-3D-texture pass with the appropriate shader enabled. This shader approximates the attenuation cone Ψ for each voxel and light source

Dataset	Volume Size	SAT Size	Shells	Cone Samples	Cone Angle	Lights [Dynamic / Static]	With Illumination	Without Illumination	Figure
Head	128 x 256 x 256	192^3	15	n/a	n/a	ambient only	111fps	143fps	2
Engine	256 x 256 x 128	64^3	n/a	50	10°	1 / 0	57fps	130fps	4
Cornell box	256 x 256 x 256	256^3	5	60	16°	0 / 1	133fps	161fps	6
Pelvis	512 x 512 x 461	64^3	5	50	2°	2 / 0	15fps	26fps	8
Feet	512 x 512 x 250	128^3	3	50	2°	1 / 0	13fps	31fps	8
Bucky ball	128 x 128 x 128	128^3	3	80	2°	0 / 1, 2, 3	55fps	62fps	1, 9
Skull	128 x 256 x 256	128^3	5	50	2°	0 / 2	14fps	26fps	9
Pelvis (comparison)	512 x 512 x 461	192^3	3	40	1°, 3°, 5°	1 / 0	3fps	30fps	10
Engine (comparison)	256 x 256 x 128	64^3, 128^3	5	40	2°	1 / 0	26fps, 12fps	49fps	11
Engine (comparison)	256 x 256 x 128	192^3, 256^3	5	40	2°	1 / 0	5fps, 3fps	49fps	11

Table 1. Overall frame rates with and without extinction-based illumination for a 512^2 pixel viewport. Except for the head, Cornell box, skull and bucky ball, the directional soft shadows are computed dynamically for every frame.

Fig. 8. Medical datasets rendered with extinction based-shading and illumination including directional soft shadows and ambient occlusion. The left image is rendered using two lights and shows multiple shadows.

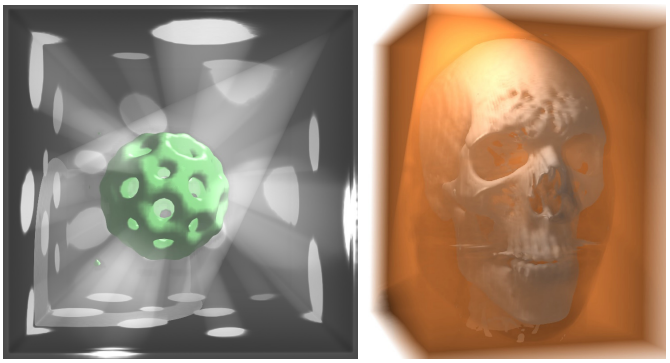

Fig. 9. Bucky ball in a smoky cube where a point light source is inside the bucky ball and a spot light in the top right corner (left), and Skull in thick fog or a block of translucent material with a point light source in the back scattering light through the medium and a spot light in the top left corner (right).

by a series of cuboids. The primary cone axis is defined to be the coordinate axis with the smallest angle to the vector to the light source. The sampling points on the primary axis are given by a user defined sampling frequency and growth rate. The growth rate (growth of the cuboids) is the change of the frequency over the distance since further away a smaller sampling frequency may be sufficient. The cuboid queries are then derived from this primary axis sampling and from the projection of the query cone onto the primary-secondary axis planes as shown in Fig. 7. Because the SAT inherently allows only axis-aligned lookups, deriving the primary and secondary axes is required. Choosing the axes in this way yields the best possible coverage of the cone with cuboids. With the cone covered by cuboids, the summation of the extinction coefficients can quickly be obtained by a few SAT lookups.

The shadow and scattering approximation by extinction SAT queries makes it very fast and flexible. The number of cuboids and the cone angle of Ψ can easily be varied, or the cuboids can be weighted differently using h in order to strengthen or weaken the effect. Approximating the cone by exploiting the SAT allows for soft, realistic looking, directional shadows at very low costs as shown in Fig. 4. In contrast to the half angle slicing method by Kniss et al. [14] our solution can handle any type and multiple light sources. It is also different from the method by Ropinski et al. [25] because we do not propagate illumination from the outside but compute the extinction of the light intensity for the voxels. We can therefore trivially handle light sources even within or on the border of the dataset without any additional effort. Multiple light sources can also be easily dealt with (see Fig. 1 for multiple point and spot light sources inside the volume).

The angle of the cone Ψ, the number of cuboids for approximation (defined by a sampling frequency), the growth rate, and a weighting function are parameters that can be chosen freely according to the desired quality/performance and strength of the shadow effects (see Fig. 10). Typically a few dozen lookups per cone and voxel are already sufficient to approximate the attenuation cone, compared to classical shadow rays where hundreds of samples are required to achieve a similar quality (see Section 4). Hence, even when computing these shadow terms for every frame, the performance impact is tolerable with respect to the achieved shading effects. To avoid duplicate shadow queries, the computed terms are stored in the light cache texture together with the terms for ambient occlusion/color bleeding.

4 Results

All experiments have been performed on a Mac Pro 2.4GHz dual-Xeon with NVIDIA GeForce GTX 285 graphics.

Compared to a Phong-Blinn-based GPU ray-caster, a ray-caster with our illumination model can produce realistic looking images with improved depth and occlusion effects (i.e. Figs. 8, 10, 11). To ensure interactivity and responsivity, we use a 3D SAT enabling fast approximation of shadow cones with cuboids and AO/color bleeding using cuboid shells. For each change of the TF, the extinction SAT and the AO/color bleeding terms have to be recomputed. Every time the light source moves relative to the dataset or the TF changes, the terms for the directional shadows will be recomputed. During the actual ray-casting pass, one additional texture lookup per sample is sufficient to apply the illumination terms. Other approaches [27] need two additional texture lookups for AO, not considering directional shadows.

Table 1 demonstrates the interactive performance of our extinction-based illumination model. This includes computation of the SAT and the ambient factors once and the factors for directional shadows in every frame. The exceptions are the head, Cornell box, skull and bucky ball datasets where the factors for the directional shadows have to be computed only once due to the fixed light source(s).

The time required for constructing the 3D SAT for different sizes is 0.029, 0.067, 0.148 and 0.311s for 64^3, 128^3, 192^3 and 256^3 volumes respectively. Even though we do not use CUDA, the time is similar to the one reported by Diaz et al. [4] for the 256^3 volume and is in fact much faster for smaller volume sizes. Moreover, our timings include the concurrent construction of the 3D SAT comprising the terms for

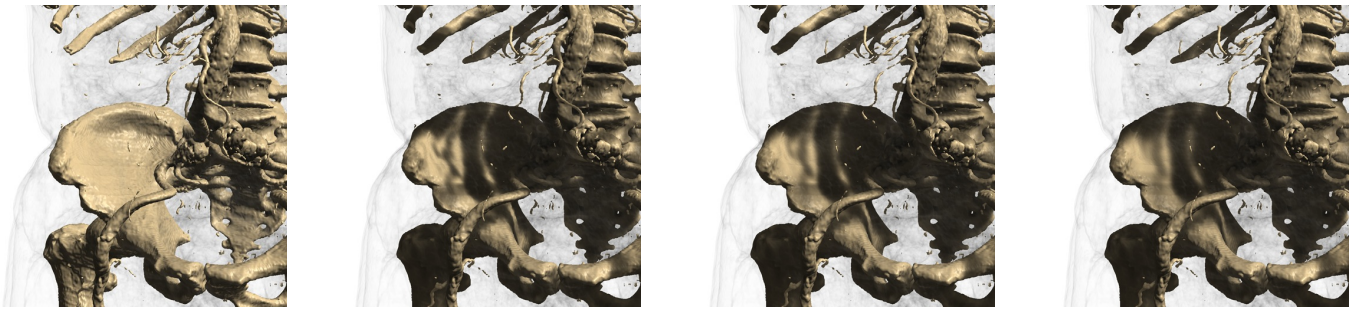

Fig. 10. Pelvis rendered with no directional shadows at all (left), and with different cone angles of 1.0, 3.0 and 5.0 degrees, causing progressively smoother shadows (from 2nd left to right).

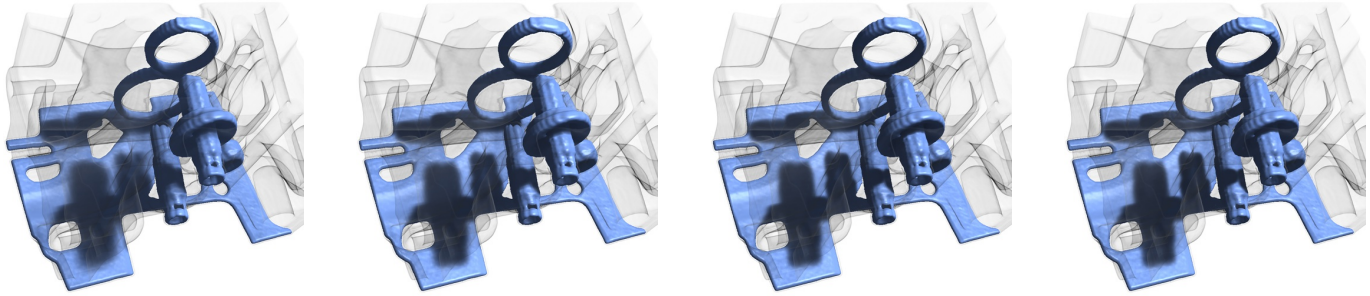

Fig. 11. The engine dataset rendered with different SAT resolutions of 64^3, 128^3, 192^3 and 256^3 and 40 cone samples (from left to right). Even lower SAT sizes will result in more blurred shadows but not expose conspicuous artifacts.

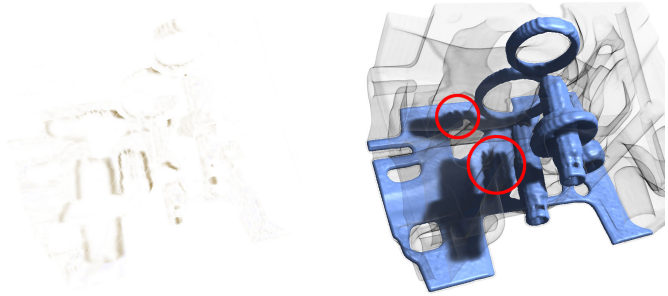

Fig. 12. As few as 12 cone samples are sufficient until cuboid artifacts become clearly visible for a SAT resolution of 128^3. The image on the left shows the difference to the respective image from Figure 11.

color bleeding. Hence we can see that even for dynamic TF changes an interactive feedback can be achieved. The influence of the SAT size on the rendering is shown in Fig. 11, demonstrating that the SAT size can easily be set at a fraction of the size of the volume dataset itself.

Table 2 shows the time required for computing the actual terms for ambient occlusion/color bleeding by approximating the neighborhood of every voxel with cuboid shells. The time is only dependent on the volume size and the number of shells but not on the neighborhood radius in contrast to explicit neighborhood sampling. Nevertheless, the last column shows the time for explicitly sampling a neighborhood of radius 7 ($= \frac{4}{3}\pi 7^3$ samples) to demonstrate the large time cost difference. Thus even if not cached, ambient occlusion/color bleeding effects can be computed in real-time for these volume models. The head in Fig. 2 was rendered using a 192^3 AO texture with 15 shells.

Note that combining SAT construction and AO computation times for an example as in Fig. 2 results in a fairly low cost of only about 0.073s. This highly interactive SAT and AO computation avoids costly preprocessing [27] to achieve TF independence.

Table 3 shows the time required for computing directional soft shadows for two different cuboidal cone approximation resolutions. The time required and the quality obviously depend on the sampling parameters. The last column shows the time for a single, classical shadow ray with a sampling rate of 250 samples per unit (the side length of volume dataset). In fact, to get equally soft shadows, the cost of a single shadow ray would have to be multiplied by a factor ($\gg 1$) because sampling would have to be performed within an entire cone and not only on the ray.

AO Shadow Texture Size	11 Shells	7 Shells	3 Shells	Sampled Radius=7
64 x 64 x 64	0.0039s	0.0030s	0.0022s	0.2099s
128 x 128 x 128	0.0202s	0.0126s	0.0065s	0.4441s
192 x 192 x 192	0.0606s	0.0348s	0.0147s	1.4280s
256 x 256 x 256	0.1382s	0.0791s	0.0307s	3.0818s

Table 2. Time needed for computing the AO/color bleeding terms for different light cache texture sizes and numbers of shells. The last column shows the time for explicitly sampling the neighborhood of radius $r = 7$.

Soft Shadow Texture Size	50 Samples	20 Samples	1 Shadow Ray
64 x 64 x 64	0.0079s	0.0061s	0.0098s
128 x 128 x 128	0.0385s	0.0270s	0.0361s
192 x 129 x 129	0.1166s	0.0816s	0.0954s
256 x 256 x 256	0.2671s	0.1821s	0.2105s

Table 3. Time required for computing directional soft shadows for different volume sizes and two different cone samplings. The last column shows the time for a single, classical shadow ray with a sampling rate of 250 samples per unit (the side length of the volume).

For the engine in Fig. 4 and the first medical dataset in Fig. 8 it is sufficient to use a 64^3 AO/shadow texture with 50 samples per unit, consuming only 0.0079s/frame for shadow computations, to achieve a very good image quality. The second medical dataset in Fig. 8 was rendered using a 128^3 AO/shadow texture with 50 samples, using 0.0385s/frame for shadow computations. The Cornell box in Fig. 6 was rendered using a 256^3 AO/shadow texture with 60 samples but updated only upon TF changes due to the fixed embedded light source. Fig. 10 shows the effects of different cone angles. Fig. 12 demonstrates the artifacts from the cuboids that become visible if only very few cone samples are used.

The scattering of light in smoke, thick fog or wax-like media with non-zero opacity is demonstrated in Fig. 9, clearly showing the expected light shafts and diffusely scattered light propagation. Multiple

different point and spot light sources inside the volume dataset and the corresponding illumination and shadow effects are demonstrated in Fig. 1 and 9. Our solution can transparently and efficiently handle any such light sources (unlike e.g. half-angle slicing).

5 Conclusion and Future Work

In this paper we have presented a novel advanced illumination and shading model for direct volume rendering based on the original exponential extinction of the volume rendering integral. The fact that the original exponential extinction is additive allows us to exploit a summed area table (SAT) concept for the efficient computation of any type of ambient or directional light occlusion queries.

In our work, ambient occlusion, color bleeding, soft shadows and scattering effects are evaluated through a series of 3D SAT queries, thus benefitting in speed while maintaining high visual quality. The method and its implementation take advantage of the SIMD architecture of OpenGL and GLSL shaders for concurrently constructing the SAT not only for extinction coefficients but also for color value aggregation at virtually no additional cost. Interactive frame rates can be achieved on scenes with static and dynamic lights as well as for dynamic transfer function changes on moderate size volume datas.

A limitation of our implementation is that specific shapes of light sources such as neon tubes are not taken into account and only basic point, spot and area light sources are supported. We would like to resolve this issue in future work.

We would also like to investigate scalability issues for large datasets that do not fit on GPU. Once the GPU memory runs low, it is common to employ bricking [6] in GPU ray-casting. Nothing prevents applying bricking to our illumination model, where the auxiliary 3D textures can either be kept resident in the GPU memory (if small enough) and only the dataset is bricked, or they can be bricked as well. We expect that the performance penalty will be proportional to bricking without our illumination model applied.

Acknowledgments

The authors wish to thank volvis.org for the engine model, the National Library of Medicine-NIH (and Univ. Erlangen) for the VisMale data, OsiriX for the chest, pelvis and feet DICOM images, and AVS (Univ. Erlangen) for the bucky ball volume data. This work was supported in part by the Swiss National Science Foundation under Grant 200020-129525/1.

References

[1] U. Behrens and R. Ratering. Adding shadows to a texture-based volume renderer. In *Proceedings IEEE Symposium on Volume Visualization*, pages 39–46, 1998.

[2] S. Bruckner and E. Groller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13:1344–1351, 2007.

[3] F. C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH, pages 207–212, 1984.

[4] J. Díaz, P.-P. Vázquez, I. Navazo, and F. Duguet. Real-time ambient occlusion and halos with summed area tables. *Computers & Graphics*, 34(4):337–350, August 2010.

[5] P. Dubois and G. Rodrigue. An analysis of the recursive doubling algorithm. In D. H. L. D. J. Kuck and A. H. Sameh, editors, *High Speed Computer and Algorithm Organization*, pages 299–305. Academic Press, 1977.

[6] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. A. K. Peters, Ltd., 2006.

[7] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, pages 9–16, 2001.

[8] C. Everitt. Interactive order-independent transparency. Technical report, NVIDIA Corp., 2001.

[9] M. Hadwiger, P. Ljung, C. Rezk-Salama, and T. Ropinski. Advanced illumination techniques for GPU-based volume raycasting. ACM SIGGRAPH Course Notes, 2009.

[10] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra. Fast summed-area table generation and its applications. *Computer Graphics Forum*, 24:547–555, 2005.

[11] F. Hernell, P. Ljung, and A. Ynnerman. Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):548–559, July/August 2010.

[12] H. W. Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., 2001.

[13] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[14] J. Kniss, S. Premoze, C. Hansen, and D. Ebert. Interactive translucent volume rendering and procedural modeling. In *Proceedings IEEE Visualization*, pages 109–116, 2002.

[15] M. Kraus and K. Bürger. Interpolating and downsampling RGBA volume data. In *Proceedings VMV*, pages 323–332, 2008.

[16] J. Kruger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proceedings IEEE Visualization*, pages 287–292, 2003.

[17] H. Landis. Production-ready global illumination. In *Siggraph Course Notes*, volume 16, 2002.

[18] M. Malmer, F. Malmer, U. Assarsson, and N. Holzschuch. Fast precomputed ambient occlusion for proximity shadows. *Journal of graphics tools*, 12(2):59–71, April 2007.

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

[20] A. Méndez, M. Sbert, and J. Catà. Real-time obscurances with color bleeding. In *Proceedings Spring Conference on Computer Graphics*, pages 171–176, 2003.

[21] À. Méndez-Feliu and M. Sbert. From obscurances to ambient occlusion: A survey. *The Visual Computer*, 25(2):181–196, February 2008.

[22] K. D. Moreland. *Fast High Accuracy Volume Rendering*. PhD thesis, The University of New Mexico, 2004.

[23] G. Papaioannou, M. L. Menexi, and C. Papadopoulos. Real-time volume-based ambient occlusion. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):752–762, September/October 2010.

[24] T. Porter and T. Duff. Compositing digital images. In *Proceedings ACM SIGGRAPH*, pages 253–259, 1984.

[25] T. Ropinski, C. Döring, and C. Rezk-Salama. Interactive volumetric lighting simulating scattering and shadowing. In *Proceedings IEEE Pacific Visualization Symposium*, pages 169–176, 2010.

[26] T. Ropinski, J. Kasten, and K. Hinrichs. Efficient shadows for GPU-based volume raycasting. In *Proceedings International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 17–24, 2008.

[27] T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. H. Hinrichs. Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum*, 27(2):567–576, 2008.

[28] M. Ruiz, I. Boada, I. Viola, S. Bruckner, M. Feixas, and M. Sbert. Obscurance-based volume rendering framework. In *Proceedings IEEE/EG Symposium on Volume and Point-Based Graphics*, pages 113–120, 2008.

[29] H. Scharsach. Advanced GPU raycasting. In *Proceedings Central European Seminar on Computer Graphics (CESCG)*, pages 69–76, 2005.

[30] P. Schlegel and R. Pajarola. Layered volume splatting. In *Proceedings International Symposium on Visual Computing*, pages 1–12, 2009.

[31] M. Schott, V. Pegoraro, C. Hansen, K. Boulanger, and K. Bouatouch. A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum*, 28(3):855–862, June 2009.

[32] P. Shanmugam and O. Arikan. Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings Symposium on Interactive 3D Graphics and Games*, pages 73–80. ACM SIGGRAPH, 2007.

[33] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A simple and flexible volume rendering framework for graphics-hardware–based raycasting. In *Proceedings International Workshop on Volume Graphics*, pages 187–195, 2005.

[34] L. Westover. Footprint evaluation for volume rendering. In *Proceedings ACM SIGGRAPH*, pages 367–376. ACM SIGGRAPH, 1990.

[35] L. Williams. Casting curved shadows on curved surfaces. In *Proceedings ACM SIGGRAPH*, pages 270–274, 1978.

[36] C. Zhang and R. Crawfis. Shadows and soft shadows with participating media using splatting. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):139–149, 2003.