State of The Art Report on Interactive Volume Rendering with Volumetric Illumination

Daniel Jönsson, Erik Sundén, Anders Ynnerman, and Timo Ropinski

Scientific Visualization Group, Linköping University, Sweden

Abstract

Interactive volume rendering in its standard formulation has become an increasingly important tool in many application domains. In recent years several advanced volumetric illumination techniques to be used in interactive scenarios have been proposed. These techniques claim to have perceptual benefits as well as being capable of producing more realistic volume rendered images. Naturally, they cover a wide spectrum of illumination effects, including varying shadowing and scattering effects. In this article, we review and classify the existing techniques for advanced volumetric illumination. The classification will be conducted based on their technical realization, their performance behavior as well as their perceptual capabilities. Based on the limitations revealed in this review, we will define future challenges in the area of interactive advanced volumetric illumination.

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

1. Introduction

Interactive volume rendering as a subdomain of scientific visualization has become mature in the last decade. Several approaches exist, which allow a domain expert to visualize and explore volumetric data sets at interactive frame rates on standard graphics hardware. While the varying rendering paradigms underlying these approaches directly influence image quality [SHC*09], in most cases the standard emission absorption model is used to incorporate illumination effects [Max95]. However, the emission absorption model is only capable of simulating local lighting effects, where light is emitted and absorpted locally at each sample point processed during rendering. This results in volume rendered images, which convey the basic structures represented in a volumetric data set (see Figure 1 *(left)*). Though, all structures are clearly visible in these images, information regarding the arrangement and size of structures is sometimes hard to convey. In the computer graphics community, more precisely the real-time rendering community, the quest for realistic interactive image synthesis is one of the major goals addressed since its early days [DK09]. Originally, mainly motivated by the goals to be able to produce more realistic imagery for computer games and animations, the perceptual benefits of more realistic representations could also be shown [WFG92, Wan92, GP06]. Consequently, in the past years the scientific visualization community started to develop interactive volume rendering techniques, which do not only allow to simulate local lighting effects, but also more realistic global effects [RSHRL09]. The existing approaches span a wide range in terms of underlying rendering paradigms, consumed hardware resources as well as lighting capabilities, and thus have different perceptual impacts [LR11]. In this article, we focus on advanced volume illumination techniques, which allow interactive data exploration, i. e., rendering parameters can be changed interactively. When developing such techniques, several challenges need to be addressed:

- A volumetric optical model must be applied, which usually results in more complex computations due to the global nature of the illumination.
- Interactive transfer function updates must be supported. This requires, that the resulting changes to the 3D structure of the data must be incorporated during illumination.
- Graphics processing unit (GPU) algorithms need to be developed to allow interactivity.

When successfully addressing these challenges, increasingly realistic volume rendered images can be generated interactively. As can be seen in Figure 1 *(right)*, these images not only increase the degree of realism, but also support improved perceptual comprehension. While Figure 1 *(left)*

DOI: 10.2312/conf/EG2012/stars/053-074

Figure 1: *Comparison of a volume rendered image using the local emission absorption model [Max95] (left), and the global half angle slicing technique [KPHE02] (right). Apart from the illumination model, all other image parameters are the same.*

shows the overall structure of the rendered computed tomography (CT) data set of a human heart, the shadows added in Figure 1 *(right)* provide additional depth information.

We will review and compare the existing techniques for advanced volumetric illumination with respect to their applicability, which we derive from the illumination capabilities, the performance behavior as well as their technical realization. Since this report addresses interactive advanced volumetric illumination techniques, potentially applicable in scientific visualization, we do not consider methods requiring precomputations that do not permit change of the transfer function during rendering. The goal is to provide the reader with an overview of the field and to support design decisions when exploiting current concepts. We have decided to fulfill this by adapting a classification, which takes into account the technical realization, performance behavior as well as the supported illumination effects. Properties considered in the classification are for instance, the usage of preprocessing, because precomputation based techniques are usually not applicable to large volumetric data sets as the precomputed illumination volume would consume too much additional graphics memory. Other approaches might be bound to a certain rendering paradigm, which serves as another property, since it might hamper usage of a technique in a general manner. We have selected the covered properties, such that the classification allows an applicability-driven grouping of the existing advanced volumetric illumination techniques. To provide the reader with a mechanism to choose which advanced volumetric illumination model to be used, we will describe these and other properties as well as their implications along with our classification. Furthermore, we will point out the shortcomings of the current approaches to demonstrate possibilities for future research.

The remainder of this article is structured as follows. In the next section we will discuss the models used for simulating volumetric illumination. We will have our starting point in the seminal work presented by Max [Max95], with the emission absorption model which we successively enrich to later on support global volumetric illumination effects. The reader interested mainly in the concepts underlying the implementations of the approaches covered in this article may skip Section 2 and directly proceed to Section 3, where we classify the covered techniques. Based on this classification, Section 4 discusses all covered techniques in greater detail. Since one of the main motivations for developing volumetric illumination models in the scientific visualization community is improved visual perception, we will discuss recent findings in this area with respect to interactive volume rendering in Section 5. By considering both, the technical capabilities of each technique as well as its perceptual outcome, we propose usage guidelines in Section 6. These guidelines have been formulated with the goal to support an application expert choosing the right algorithm for a specific use case scenario. Furthermore, based on the observations made in Section 5 and the limitations identified throughout the article, we will present future challenges in Section 7. Finally, the article concludes in Section 8.

2. Volumetric Illumination Model

In this section we derive the volume illumination model frequently exploited by the advanced volume illumination approaches described in literature. The model is based on the optical model derived by Max [Max95] as well as the extensions more recently described by Max and Chen [MC10]. For clarity, we have included the definitions used within the model as a reference below.

Mathematical Notation

$L_s(\vec{x},\vec{\omega}_o)$	Radiance scattered from $\vec{x}$ into direction $\vec{\omega}_o$.
$L_i(\vec{x},\vec{\omega}_i)$	Radiance reaching $\vec{x}$ from direction $\vec{\omega}_i$.
$L_e(\vec{x})$	Radiance isotropically emitted from $\vec{x}$.
$s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$	Shading function used at position $\vec{x}$. Similar to a BRDF, it is dependent on the incoming light direction $\vec{\omega}_i$ and the outgoing light direction $\vec{\omega}_o$.
$p(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$	Phase function used at position $\vec{x}$. As the shading function s, it is dependent on the incoming light direction $\vec{\omega}_i$ and the outgoing light direction $\vec{\omega}_o$.
$\tau(\vec{x})$	Extinction occurring at $\vec{x}$.
$T(\vec{x}_l,\vec{x})$	Transparency between the light source position $\vec{x}_l$ and the current position.
L_l	Initial radiance as emitted by the light source.
$L_0(\vec{x}_0,\vec{\omega}_o)$	Background intensity.

Similar to surface-based models, in volume rendering the lighting is computed per point, which is in our case a sample $\vec{x}$ along a viewing ray. The radiance $L_s(\vec{x},\vec{\omega}_o)$ which is scattered from $\vec{x}$ inside the volume into direction $\vec{\omega}_o$ can be defined as:

$$L_s(\vec{x},\vec{\omega}_o) = s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)\cdot L_i(\vec{x},\vec{\omega}_i) + L_e(\vec{x}), \qquad (1)$$

where $L_i(\vec{x},\vec{\omega}_i)$ is the incident radiance reaching $\vec{x}$ from direction $\vec{\omega}_i$, $L_e(\vec{x})$ is the emissive radiance and $s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ describes the actual shading, which is dependent on both the incident light direction $\vec{\omega}_i$ as well as the outgoing light direction $\vec{\omega}_o$. Furthermore, $s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ is dependent on parameters, which may vary based on $\vec{x}$, as for instance the optical properties assigned through the transfer function. In the context of volume rendering $s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ is often written as:

$$s(\vec{x},\vec{\omega}_i,\vec{\omega}_o) = \tau(\vec{x}) \cdot p(\vec{x},\vec{\omega}_i,\vec{\omega}_o), \tag{2}$$

where $\tau(\vec{x})$ represents the extinction coefficient at position $\vec{x}$ and $p(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ is the phase function describing the scattering characteristics of the participating medium at the position $\vec{x}$.

Scattering is the physical process which forces light to deviate from its straight trajectory. The reflection of light at a surface point is thus a scattering event. Depending on the material properties of the surface, incident photons are scattered in different directions. When using the ray-casting analogy, scattering can be considered as a redirection of a ray penetrating an object. Since scattering can also change a photons frequency, a change in color becomes possible. Max [Max95] presents accurate solutions for simulating light scattering in participating media, i. e., how the light trajectory is changed when penetrating translucent materials. In classical computer graphics, single scattering accounts for light emitted from a light source directly onto a surface and reflected unimpededly into the observer's eye. Incident light thus comes directly from a light source. Multiple scattering describes the same concept, but incoming photons are scattered multiple times. To generate realistic images, both *indirect light* and *multiple scattering events* have to be taken into account. Both concepts are closely related, they only differ in their scale. Indirect illumination means that light is reflected multiple times by different objects in the scene. Multiple scattering refers to the probabilistic characteristics of a scattering event caused by a photon being reflected multiple times within an object. While in classical polygonal computer graphics, light reflection on a surface is often modeled using bidirectional reflectance distribution functions (BRDFs), in volume rendering phase functions are used to model the scattering behavior. Such a phase function can be thought of as being the spherical extension of a hemispherical BRDF. It defines the probability of a photon changing its direction of motion by an angle of θ. In interactive volume rendering, the Henyey-Greenstein model $G(\theta,g) = \frac{1-g^2}{(1+g^2-2g\cos\theta)^{\frac{3}{2}}}$ is often used to incorporate phase functions. The parameter $g \in [-1,1]$ describes the anisotropy of the scattering event. A value $g = 0$ denotes that light is scattered equally in all directions. A positive value of g increases the probability of forward scattering. Accordingly, with a negative value backward scattering will become more likely. If $g = 1$, a photon will always pass through the point unaffectedly. If $g = -1$ it will deterministically be reflected into the direction it came from. Figure 2 shows the Henyey-Greenstein phase function model for different anisotropy parameters g.

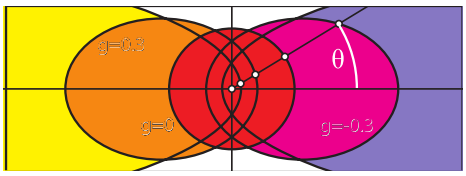

Figure 2: *The Henyey-Greenstein phase function plotted for different anisotropy parameters g.*

When considering shadowing in this model, where the attenuation of external light traveling through the volume is incorporated, the incident radiance $L_i(\vec{x},\vec{\omega}_i)$ can be defined as:

$$L_i(\vec{x},\vec{\omega}_i) = L_l \cdot T(\vec{x}_l,\vec{x}). \tag{3}$$

This definition is based on the standard volume rendering integral, where the light source is located at $\vec{x}_l$ and has the radiance L_l and $T(\vec{x}_l,\vec{x})$ is dependent on the transparency between $\vec{x}_l$ and $\vec{x}$. It describes the compositing of the optical properties assigned to all samples $\vec{x}'$ along a ray and it incorporates emission and absorption, such that it can be written as:

$$L(\vec{x},\vec{\omega}_o) = L_0(\vec{x}_0,\vec{\omega}_o) \cdot T(\vec{x}_0,\vec{x}) + \int_{\vec{x}_0}^{\vec{x}} o(\vec{x}') \cdot T(\vec{x}',\vec{x}) d\vec{x}'. \tag{4}$$

While $L_0(\vec{x}_0,\vec{\omega}_o)$ depicts the background energy entering the volume, $o(\vec{x}')$ defines the optical properties at $\vec{x}'$, and $T(\vec{x}_i,\vec{x}_j)$ is dependent on the optical depth and describes how light is attenuated when traveling through the volume:

$$T(\vec{x}_i,\vec{x}_j) = e^{-\int_{\vec{x}_i}^{\vec{x}_j} \kappa(\vec{x}') d\vec{x}'}, \tag{5}$$

where $\kappa(\vec{x}')$ specifies the absorption at $\vec{x}'$. Since volumetric data is discrete, the volume rendering integral is in practice approximated numerically, usually by exploiting a Riemann sum.

According to Max, the standard volume rendering integral simulating absorption and emission can be extended with these definitions to support single scattering and shadowing:

$$L(\vec{x},\vec{\omega}_o) = L_0(\vec{x}_0,\vec{\omega}_o)\cdot T(\vec{x}_0,\vec{x}) + \int_{\vec{x}_0}^{\vec{x}} (L_e(\vec{x}) + (s(\vec{x}',\vec{\omega}_i,\vec{\omega}_o)\cdot L_i(\vec{x}',\vec{\omega}_i)))\cdot T(\vec{x}',\vec{x})d\vec{x}', \quad (6)$$

with $\vec{\omega}_i = \vec{x}_l - \vec{x}'$, $L_0(\vec{x}_0,\vec{\omega}_o)$ being the background intensity and $\vec{x}_0$ a point behind the volume. When extending this equation to support multiple scattering, it is necessary to integrate the scattering of light coming from all possible directions $\vec{\omega}_i$ on the unit sphere.

3. Algorithm Classification

To be able to give a comprehensive overview of current interactive advanced volumetric illumination techniques, we will introduce a classification within this section, which allows us to group and compare the covered approaches. The classification has been developed with the goal to provide application designers with decision support, when choosing advanced volumetric illumination techniques for their needs. Thus, we will first cover the most relevant algorithm properties, that need to be considered when making such a decision.

The most limiting property of advanced volumetric illumination algorithms is the dependence on an underlying rendering paradigm. In the past, different paradigms allowing interactive volume rendering have been proposed. These paradigms range from a shear-warp transformation [LL94], over splatting [MMC99], and texture slicing [CCF94] to volume ray-casting [KW03]. Besides their different technical realizations, the visual quality of the rendered image also varies with respect to the used rendering paradigm [SHC*09]. Since efficiency is of great importance in the area of interactive volumetric illumination, many approaches have been developed which allow a performance gain by being closely bounded to a specific rendering paradigm (e. g., [KPHE02, ZC02, ZC03, SPH*09, vPBV10, SYR11]). Especially when integrating volumetric illumination into existing visualization systems, the underlying rendering paradigm of the existing approaches might be limiting when deciding which algorithm to be used. Therefore, we will discuss this dependency when presenting the covered algorithms in Section 4.

Another property of major importance is, which illumination effects are supported by a certain algorithm. This is on the one hand specified based on the supported lighting and on the other hand on the light interaction and propagation inside the volume. The supported illumination can vary with respect to the number and types of the light sources. Supported light source types range from classical point and directional light sources, over area and textured light sources. Since several algorithms focus on ambient visibility computation, we consider also an ambient light source which is omni directional and homogeneous. As some algorithms are bound to certain rendering paradigms, they may also be subject to constraints regarding the light source position,e. g., [BR98, SPH*09, vPBV10]. Finally, the number of supported light sources varies, where either one or many light sources are supported. With respect to the light interaction and propagation inside the volume, the discussed algorithms also vary. While all algorithms support either local or global shadowing, though at different frequencies reaching from soft to hard shadows, scattering is not supported by all approaches. Those algorithms, which support scattering may either support the simulation of single or multiple scattering events. We will discuss these capabilities for each covered technique and relate the supported illumination effects to Max's optical models [Max95].

As in many other fields of computer graphics, applying advanced illumination techniques in volume rendering involves a trade off between rendering performance and memory consumption. The two most extreme cases on this scale are entire precomputation of the illumination information, and recomputation on the fly for each frame. We will compare the techniques covered within this article with respect to this trade off, by describing the performance impact of these techniques as well as the required memory consumption. Since different application scenarios vary with respect to the degree of interactivity, we will review their performance capabilities with respect to rendering and illumination update. Some techniques trade illumination update times for frame rendering times and thus allow higher frame rates, as long as no illumination critical parameter has been changed. We discuss rendering times as well as illumination update times, whereby we differentiate whether they are triggered by lighting or transfer function updates. Recomputations performed when the camera changes are considered as rendering time. The memory footprint of the covered techniques is also an important factor, since it is often limited by the available graphics memory. Some techniques precompute an illumination volume, e. g., [RDRS10b, SMP11], while others store much less or even no illumination data, e. g., [KPHE02, DEP05, SYR11].

Finally, it is important if a volumetric illumination approach can be combined with clipping planes and allows to combine geometry and volumetric data. While clipping is frequently used in many standard volume rendering applications, the integration of polygonal geometry data is for instance important in flow visualization, when integrating pathlines in a volumetric data set.

To enable easier comparison of the existing techniques and the capabilities described above, we have decided to classify them into five groups based on the algorithmic concepts exploited to achieve the resulting illumination effects. The thus obtained groups are, first *local region based* ( ) techniques which consider only the local neighborhood around a voxel, second *slice based* ( ) techniques which

Figure 3: *Visual comparison when applying different volumetric illumination models. From left to right: gradient-based shading [Lev88], directional occlusion shading [SPH*09], image plane sweep volume illumination [SYR11], shadow volume propagation [RDRS10b] and spherical harmonic lighting [KJL*12]. Apart from the used illumination model, all other rendering parameters are constant.*

propagate illumination by iteratively slicing through the volume, third, *light space based* ( ) techniques which project illumination as seen from the light source, fourth *lattice based* ( ) techniques which compute the illumination directly on the volumetric data grid without applying sampling, and fifth *basis function based* ( ) techniques which use a basis function representation of the illumination information. While this classification into five groups is solely performed based on the concepts used to compute the illumination itself, and not for the image generation, it might in some cases go hand in hand, e. g., when considering the *slice based* techniques which are also bound to the slice based rendering paradigm. A comparison of the visual output of one representative technique of each of the five groups is shown in Figure 3. As it can be seen, when changing the illumination model, the visual results might change drastically, even though most of the presented techniques are based on Max's formulation of a volumetric illumination model [Max95]. The most prominent visual differences are the intensities of shadows as well as their frequency, which is visible based on the blurriness of the shadow borders. Apart from the used illumination model, all other rendering parameters are the same in the shown images. Besides the five main groups supporting fully interactive volume rendering, we will also briefly cover *ray-tracing based* ( ) techniques, and those only limited to *isosurface illumination*. While the introduced groups allow a sufficient classification for most available techniques, it should also be mentioned, that some approaches could be classified into more than one group. For instance, the shadow volume propagation approach [RDRS10b], which we have classified as lattice-based, could also be classified as light space based, since the illumination propagation is performed based on the current light source position.

To be able to depict more details for comparison reasons, we will cover the properties of the most relevant techniques in Table 1 at the end of this article. The table shows the properties with respect to supported light sources, scattering capabilities, render and update performance, memory consumption, and the expected impact of a low signal to noise ratio. For rendering performance and memory consumption, the filled circles represent the quantity of rendering time or memory usage. An additional icon represents, if illumination information needs to be updated when changing the lighting or the transfer function. Thus, the algorithm performance for frame rendering and illumination updates become clear.

4. Algorithm Description

In this section, we will describe all techniques covered within this article in a formalized way. The section is structured based on the groups introduced in the previous section. For each group, we will provide details regarding the techniques themselves with respect to their technical capabilities as well as the supported illumination effects. All of these properties are discussed based on the property description given in the previous section. We will start with a brief explanation of each algorithm, where we relate the exploited illumination computations to Max's model (see Section 2) and provide a conceptual overview of the implementation. Then, we will discuss the illumination capabilities with respect to supported light sources and light interactions inside the volume, before addressing performance impact and memory consumption.

4.1. Local Region Based Techniques ( )

As the name implies, the techniques described in this subsection perform the illumination computation based on a lo-

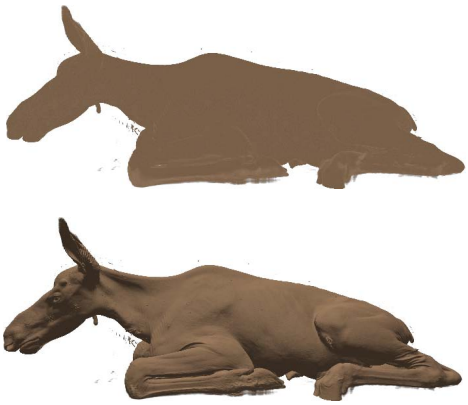

Figure 4: *A CT scan of an elk rendered with emission and absorption only (top), and with gradient-based diffuse shading [Lev88] (bottom). Due to the incorporation of the gradient, surface structures emerge (bottom).*

cal region. Thus, when relating volumetric illumination techniques to conventional polygonal computer graphics, these techniques could be best compared to local shading models. While the still most frequently used gradient-based shading [Lev88] is exploiting the concepts underlying the classical Blinn-Phong illumination model [Bli77], other techniques are more focused on the volumetric nature of the data to be rendered. Also with respect to the locality, the described techniques vary. The gradient-based shading relies on a gradient and thus takes, when for isntance using finite difference gradient operators, only adjacent voxels into account. Other techniques, as for instance dynamic ambient occlusion [RMSD*08], take into account bigger, though still local, regions. Thus, by using local region based techniques, only direct illumination is computed, i. e., local illumination not influenced by other parts of the scene. Hence, not every other part of the scene has to be considered when computing the illumination for the current object and the rendering complexity is reduced from $O(n^2)$ to $O(n)$, with n being the number of voxels.

Gradient-based volumetric shading [Lev88] is today still the most widely used shading technique in the context of interactive volume rendering. It exploits voxel gradients as surface normals to calculate local lighting effects. The actual illumination computation is done in a similar way as when using the Blinn-Phong illumination model [Bli77] in polygonal-based graphics, whereas the surface normal is substituted by the gradient derived from the volumetric data. Figure 4 shows a comparison of gradient-based volumetric shading and the application of the standard emission and absorption model. As it can be seen, surface details become more prominent when using gradient-based shading. The local illumination at a point $\vec{x}$ is computed as the sum of the three supported reflection contributions: diffuse reflection, specular reflection and ambient lighting. Diffuse and specular reflections both depend on the normalized gradient $|\bigtriangledown\tau(f(\vec{x}))|$, where $f(\vec{x})$ is the intensity value given at position $\vec{x}$. Thus, we can define the diffuse reflection as:

$$L_{diff}(\vec{x},\vec{\omega}_i) = L_l \cdot k_d \cdot max(|\bigtriangledown\tau(f(\vec{x}))| \cdot \vec{\omega}_i, 0). \quad (7)$$

Thus, we can modulate the initial diffuse lighting L_l based on its incident angle and the current voxel's diffuse color k_d. In contrast to diffuse reflections, specular reflections also depend on the viewing angle, and thus the outgoing light direction $\vec{\omega}_o$. Therefore, $\vec{\omega}_o$ is also used to modulate the initial light intensity L_l and the voxel's specular color k_s:

$$L_{spec}(\vec{x},\vec{\omega}_i,\vec{\omega}_o) = L_l \cdot k_s \cdot max(|\bigtriangledown\tau(f(\vec{x}))| \cdot \frac{\vec{\omega}_i + \vec{\omega}_o}{2}, 0)^{\alpha}. \quad (8)$$

α is used to influence the shape of the highlight seen on surface-like structures. A rather large α results in a small sharp highlight, while a smaller α results in a bigger smoother highlight. To approximate indirect illumination effects, usually an ambient term is added. This ensures, that voxels with gradients pointing away from the light source do not appear pitch black. Since this ambient term does not incorporate any spatial information, it is the easiest to compute as:

$$L_{amb}(\vec{x}) = L_l \cdot k_a, \quad (9)$$

by only considering the initial light intensity L_l and the voxels ambient color k_a. More advanced techniques, which substitute $L_{amb}(\vec{x})$ with the ambient occlusion at position $\vec{x}$, are covered below. To incorporate gradient-based shading into the volume rendering integral, the shading function $s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ is replaced by:

$$s(\vec{x},\vec{\omega}_i,\vec{\omega}_o) = \tau(\vec{x}) \cdot (L_{diff}(\vec{x},\vec{\omega}_i) + L_{spec}(\vec{x},\vec{\omega}_i,\vec{\omega}_o) + L_{amb}(\vec{x})). \quad (10)$$

To also incorporate attenuation based on the distance between $\vec{x}$ and $\vec{x}_l$, the computed light intensity can be modulated. However, this distance based weighting does not incorporate any voxels possibly occluding the way to the light source, which would result in shadowing. To solve this issue, volume illumination techniques not being constrained to a local region need to be applied.

As the gradient $\bigtriangledown\tau(f(\vec{x}))$ is used in the computation of $L_{diff}(\vec{x},\vec{\omega}_i)$ and $L_{spec}(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$, its quality is crucial for the visual quality of the rendering. Especially when dealing with a high degree of specularity, an artifact free image can only be generated when the gradients are continuous along a

Figure 5: *Gradient-based shading applied to a magnetic resonance imaging (MRI) data set (left). As it can be seen in the closeup, noise is emphasized in the rendering (right).*

boundary. However, often local difference operators are used for gradient computation, and thus the resulting gradients are subject to noise. Therefore, several approaches have been developed which improve the gradient quality. Since these techniques are beyond the scope of this article, we refer to the work presented by Hossain et al. [HAM11] as a starting point.

Gradient-based shading can be combined with all existing volume rendering paradigms, and it supports an arbitrary number of point and directional light sources. Due to the local nature, no shadows and scattering effects are supported. The rendering time, which depends on the used gradient computation scheme, is in general quite low. To further accelerate the rendering performance, gradients can also be precomputed and stored in a gradient volume accessed during rendering. When this precomputation is not performed, no additional memory is required. Clipping is directly supported, though the gradient along the clipping surfaces needs to be replaced with the clipping surface normal to avoid rendering artifacts [HLY09]. Besides the local nature of this approach, the gradient computation makes gradient-based shading highly dependent on the SNR of the rendered data. Therefore, gradient-based shading is not recommendet for modalities suffering from a low SNR, such as magnetic resonance imaging (MRI)(see Figure 5), 3D ultrasound or seismic data [PBVG10].

Local Ambient Occlusion [HLY07] is a local approximation of ambient occlusion, which considers the voxels in a neighborhood around each voxel. Ambient occlusion goes back to the obscurance rendering model [ZIK98], which relates the luminance of a scene element to the degree of occlusion in its surroundings. To incorporate this information into the volume rendering equation, Hernell et al. replace the standard ambient term $L_{amb}(\vec{x})$ by:

$$L_{amb}(\vec{x}) = \int_{\Omega}\int_{\delta(\vec{x},\omega_i,R_\Omega)}^{\delta(\vec{x},\omega_i,a)} g_A(\vec{x'}) \cdot T(\vec{x'},\delta(\vec{x},\omega_i,a))d\vec{x'}d\omega_i, \tag{11}$$

where $\delta(\vec{x},\omega_i,a)$ offsets the position by a along the direction ω_i, to avoid self occlusion. R_Ω specifies the radius of the local neighborhood for which the occlusion is computed, and $g_a(\vec{x})$ denotes the light contribution at each position $\vec{x}$ along a ray. Since the occlusion should be computed in an ambient manner, it needs to be integrated over all rays in the neighborhood Ω around a position $\vec{x}$.

Different light contribution functions $g_(\vec{x})$ are presented. To achieve sharp shadow borders, the usage of a Dirac delta is proposed, which ensures that light contributes only at the boundary of Ω. Smoother results can be achieved, when the ambient light source is considered volumetric. This results in adding a fractional light emission at each sample point:

$$g_A(\vec{x}) = \frac{1}{R_\Omega - a}. \tag{12}$$

Reformulation of $g_A(\vec{x})$ also allows to include varying emissive properties for different tissue types. When such an emissive mapping $L'_e(\vec{x})$ is available, $g_A(\vec{x})$ can be defined as follows:

$$g_A(\vec{x}) = \frac{1}{R_\Omega - a} + L'_e(\vec{x}). \tag{13}$$

An application of the occlusion only illumination is shown in Figure 6 *(left)*, while Figure 6 *(right)* shows the application of the additional emissive term, which is set proportional to the signal strength of a coregistered functional magnetic resonance imaging (fMRI) signal [NEO*10].

A straight forward implementation of local ambient occlusion would result in long rendering times, as for each $\vec{x}$ multiple rays need to be cast to determine the occlusion factor. Therefore, it has been implemented by exploiting a multiresolution approach [LLY06]. This flat blocking data structure represents different parts of the volume at different levels of detail. Thus, empty homogeneous regions can

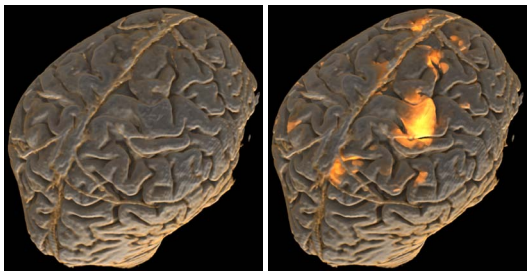

Figure 6: *An application of local ambient occlusion in a medical context. Usage of the occlusion term allows to convey the 3D structure (left), while multimodal emissive lighting allows to integrate fMRI as additional modality (right). (images taken from [NEO*10])*

be stored at a very low level of detail, which requires less sampling operations when computing $L_{amb}(\vec{x})$.

Local ambient occlusion is not bound to a specific rendering paradigm. When combined with gradient-based lighting, it supports point and directional light sources, while the actual occlusion simulates an exclusive ambient light only. The rendering time is interactive, while the multiresolution data structure needs to be updated whenever the transfer function has been changed. The memory size of the multiresolution data structure depends on the size and homogeneity of the data set. In a follow up work, more results and the usage of clipping planes during the local ambient occlusion computation are discussed [HLY09].

Ruiz et al. also exploit ambient illumination in their obscurance based volume rendering framework [RBV*08]. To compute the ambient occlusion, Ruiz et al. perform a visibility test along the occlusion rays and evaluate different distance weightings. Besides advanced illumination, they also facilitate the occlusion to obtain illustrative renderings as well as for the computation of saliency maps.

Dynamic Ambient Occlusion [RMSD*08] is a histogram based approach, which allows integration of ambient occlusion, color bleeding and basic scattering effects, as well as a simple glow effect that can be used to highlight regions within the volume. While the techniques presented in [HLY07] and [HLY09] exploit a ray-based approach to determine the degree of occlusion in the neighborhood of $\vec{x}$, dynamic ambient occlusion uses a precomputation stage, where intensity distributions are computed for each voxel's neighborhood. The main idea of this approach is to modulate these intensity distributions with the transfer function during rendering, such that an integration over the modulated result allows to obtain the occlusion. The intensity distributions are stored as normalized local histograms. When storing one normalized local histogram for each voxel, this would result in a large amount of data which needs to be stored and accessed during rendering. Therefore, a similarity based clustering is performed on the histograms, whereby a vector quantization approach is exploited. After clustering, the local histograms representing a cluster are available as a 2D texture table, which is accessed during rendering. Additionally, a scalar volume data set is generated, which contains a cluster ID for each voxel, which associates the voxel with the corresponding row storing the local histogram in the 2D texture.

During rendering, the precomputed information is used to support four different illumination effects: ambient occlusion, color bleeding, scattering and a simple glow effect. Therefore, the representative histogram is looked up for the current voxel and then modulated with the transfer function color to produce the final color. To integrate ambient occlusion into an isosurface based volume renderer, the gradient-based shading formulation described above is modulated, such that the ambient intensity $L_{amb}(\vec{x})$ is derived from the precomputed data:

$$L_{amb}(\vec{x}) = 1.0 - O_{env}(\vec{x}, \bigtriangledown\tau(f(\vec{x}))) \cdot k_a, \tag{14}$$

where O_{env} is the local occlusion factor, which is computed as:

$$O_{env}(\vec{x}, \bigtriangledown\tau(f(\vec{x}))) = \frac{1}{\frac{2}{3}\pi{\Omega_R}^3} \sum_{0 \le j < 2^b} \tau_\alpha(j) \cdot LH_j(\vec{x}). \tag{15}$$

Here, Ω_R depicts the radius of the region, for which the local histograms LH have been precomputed. b is the data bit depth and thus 2^b is the number of bins of the local histogram. The bins LH_j are modulated based on the transparency $\tau_\alpha(j)$ assigned to each intensity j. This operation is performed through texture blending on the GPU, and thus allows to update the occlusion information in real-time. To apply this approach to direct volume rendering, the environmental color E_{env} is derived in a similar way as the occlusion factor O_{env}:

$$E_{env}(\vec{x}, \bigtriangledown\tau(f(\vec{x}))) = \frac{1}{\frac{2}{3}\pi{\Omega_R}^3} \sum_{0 \le j < 2^b} \tau_\alpha(j) \cdot \tau_{rgb}(j) \cdot LH_j(\vec{x}), \tag{16}$$

where $\tau_{rgb}(j)$ represents the emissive color of voxels having the intensity j. During rendering, the current voxel's diffuse color coefficient k_d is obtained by blending between the transfer function color τ_{rgb} and E_{env} with using the occlusion factor O_{env} as blend factor. Thus, color bleeding is simulated, as the current voxel's color is affected by its environment. To support a simple glow effect, an additional mapping function can be exploited, which is also used to modulate the local histograms stored in the precomputed 2D texture.

Dynamic ambient occlusion is not bound to a specific rendering paradigm, though in the original paper it has been introduced in the context of volume ray-casting [RMSD*08]. Since it facilitates a gradient-based Blinn-Phong illumination model, which is enriched by ambient occlusion and color bleeding, it supports point, distant and ambient light sources. While an ambient light source is always exclusive, the technique can be used with several point and distant light sources, which are used to compute L_{diff} and L_{spec}. Since the precomputed histograms are constrained to local regions, only local shadows are supported, which have a soft appearance. A basic scattering extension, exploiting one histogram for each halfspace defined by a voxel's gradient, is also supported. The rendering performance is interactive, as in comparison to standard gradient-based shading only two additional texture lookups need to be performed. The illumination update time is also very low, since the histogram modulation is performed through texture blending. For the

precomputation stage, Ropinski et al. [RMSD*08] report expensive processing times, which are required to compute the local histograms and to perform the clustering. In a more recent approach, Mess and Ropinski propose a CUDA-based approach which reduces the precomputation times by a factor of up to 10 [MR10]. During precomputation, one additional scalar cluster lookup volume as well as the 2D histogram texture are generated. Since the histogram computation takes a lot of processing time, interactive clipping is not supported, as it would affect the density distribution in the voxel neighborhoods.

4.2. Slice Based Techniques (■)

The slice based techniques covered in this subsection are all bound to the slice based rendering paradigm, originally presented by Cabral et al. [CCF94]. This volume rendering paradigm exploits a stack consisting of a high number of rectangular polygons, which are used as a proxy geometry to represent a volumetric data set. Different varieties of this paradigm exist, though today image plane aligned polygons, to which the volumetric data set stored in a 3D texture is mapped, are most frequently used.

Half Angle Slicing was introduced by Kniss et al. [KPHE02]. It is the first of the slice based approaches, which synchronizes the slicing used for rendering with the illumination computation, and the first integration of volumetric illumination models including indirect lighting and scattering within interactive applications. The indirect lighting is modeled as:

$$L_i(\vec{x},\vec{\omega}_i) = L_l \cdot T(\vec{x}_l,\vec{x}). \tag{17}$$

To include multiple scattering, a directed diffusion process is modeled through an angle dependent blur function. Thus, the indirect lighting is rewritten as:

$$L_i(\vec{x},\vec{\omega}_i) = L_l \cdot T(\vec{x}_l,\vec{x}) + L_l \cdot T(\vec{x}_l,\vec{x}) Blur(\theta), \tag{18}$$

where θ is a user defined cone angle. In addition to this modified indirect lighting, which simulates multiple scattering, half angle slicing also exploits a surface shading factor $S(\vec{x})$. $S(\vec{x})$ can either be user defined through a mapping function or derived from the gradient magnitude at $\vec{x}$. It is used to weight the degree of emission and surface shading, such that the illumination $C(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ at $\vec{x}$ is computed as:

$$C(\vec{x},\vec{\omega}_i,\vec{\omega}_o) = L_e(\vec{x}) \cdot (1 - S(\vec{x})) + s(\vec{x},\vec{\omega}_i,\vec{\omega}_o) \cdot S(\vec{x}). \tag{19}$$

$C(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ is then used in the volume rendering integral as follows:

$$\begin{aligned} L(\vec{x},\vec{\omega}_o) =& L_0(\vec{x}_0,\vec{\omega}_o) \cdot T(\vec{x}_0,\vec{x}) \\ &+ \int_{\vec{x}_0}^{\vec{x}} C(\vec{x}',\vec{\omega}_i,\vec{\omega}_o) \cdot L_i(\vec{x}',\vec{\omega}_i) \cdot T(\vec{x}',\vec{x}) d\vec{x}'. \end{aligned} \tag{20}$$

In a follow up work, Kniss et al. have proposed how to enhance their optical model by using phase functions [KPH*03]. Therefore, the indirect lighting contribution is rewritten as:

$$L_i(\vec{x},\vec{\omega}_i) = L_l \cdot p(\vec{x},\vec{\omega}_i,\vec{\omega}_o) \cdot T(\vec{x}_l,\vec{x}) + L_l \cdot T(\vec{x}_l,\vec{x}) Blur(\theta). \tag{21}$$

The directional nature of the indirect lighting in $L_i(\vec{x},\vec{\omega}_i)$ supports an implementation, where the rendering and the illumination computation are synchronized. Kniss et al. achieve this by exploiting the half angle vector introduced before [KKH02]. This technique choses the slicing axis, such that it is halfway between the light direction and the view direction, or the light direction and the inverted view direction, if the light source is behind the volume. Thus, each slice can be rendered from the point of view of both the observer and the light, and no intermediate storage for the illumination information is required. The presented implementation uses three buffers for accumulating the illumination information and compositing along the view ray.

Due to the described synchronization behavior, half angle slicing is, as all techniques described in this subsection, bound to the slicing rendering paradigm. It supports one directional or point light source located outside the volume, and allows to produce shadows having hard, well defined borders (see Figure 1 *(right)*). By integrating a directed diffusion process in the indirect lighting computation, multiple scattering can be simulated which allows a realistic representation of translucent materials. Half angle slicing performs two rendering passes for each slice, one from the point of view of the observer and one from that of the light source, and thus allows to achieve interactive frame rates. Since rendering and illumination computation are performed in a lockstep manner, besides the three buffers used for compositing, no intermediate storage is required. Clipping planes can be applied interactively.

Directional Occlusion Shading [SPH*09] is another slice based technique that exploits front-to-back rendering of view aligned slices. In contrast to the half angle slicing approach discussed above, directional occlusion shading does not adapt the slicing axis with respect to the light position. Instead, it solely uses a phase function $p(\vec{x},\vec{\omega}_i,\vec{\omega})$ to prop-

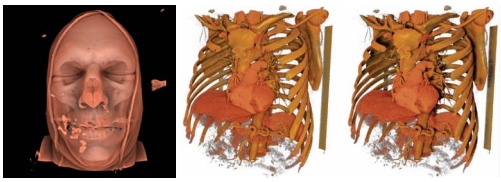

Figure 7: *Directional occlusion shading [SPH*09] integrates soft shadowing effects (left). Multidirectional occlusion shading [vPBV10] additionally supports different lighting positions (middle) and (right). (Images taken from [SPH*09] and [vPBV10], courtesy of Mathias Schott and Veronika Šoltészová)*

agate occlusion information during the slice traversal from front to back:

$$p(\vec{x},\vec{\omega}_i,\vec{\omega}) = \left\{ \begin{array}{ll} 0 & \text{if } \vec{\omega}_i \cdot \vec{\omega} < \theta(\vec{x}) \\ \frac{1}{2\pi\cdot(1-\cos(\theta))} & \text{otherwise.} \end{array} \right\}, \quad (22)$$

where $\vec{\omega}$ is the viewing direction and the $\vec{\omega}_i$'s are lying in the cone, which points with the tip along $\vec{\omega}$. Similar to the approach by Kniss et al. [KPH*03] this is a, in this case backward, peaked phase function, where the scattering direction is defined through the cone angle θ. This backward peaked behavior is important in order to support synchronized occlusion computation and rendering. Thus, $p(\vec{x},\vec{\omega}_i,\vec{\omega})$ can be used during the computation of the indirect lighting as follows:

$$L_i(\vec{x},\vec{\omega}) = L_0(\vec{x_0},\vec{\omega}) \cdot \int_\Omega p(\vec{x},\vec{\omega}_i,\vec{\omega}) \cdot T(\vec{x_l}',\vec{x}) d\vec{\omega}_i, \quad (23)$$

where the background intensity is modulated by $p(\vec{x},\vec{\omega}_i,\vec{\omega})$, which is evaluated over the contributions coming from all directions $\vec{\omega}_i$ over the sphere. Since $p(\vec{x},\vec{\omega}_i,\vec{\omega})$ is defined as a backward peaked phase function, in practice the integral needs to be only evaluated in an area defined by the cone angle θ. To incorporate the occlusion based indirect lighting into the volume rendering integral, Schott et al. propose to modulate it with a user defined scattering coefficient $\sigma_s(\vec{x})$:

$$\begin{aligned} L(\vec{x},\vec{\omega}_o) =& L_0(\vec{x_0},\vec{\omega}_o) \cdot T(\vec{x_0},\vec{x}) \\ &+ \int_{\vec{x_0}}^{\vec{x}} \sigma_s(\vec{x}') \cdot L_i(\vec{x}',\vec{\omega}_i,\vec{\omega}) \cdot T(\vec{x}',\vec{x}) d\vec{x}'. \end{aligned} \quad (24)$$

The implementation of occlusion based shading is similar to standard front-to-back slice rendering. However, similar as in half angle slicing [KPHE02] two additional occlusion buffers are used for compositing the lighting contribution.

Due to the fixed compositing order, directional occlusion shading is bound to the slice based rendering paradigm and supports only a single light source, which is located at the camera position. The iterative convolution allows to generate soft shadowing effects, and incorporating the phase function accounts for first order scattering effects (see Figure 7 *(left)* and *(middle)*). The performance is interactive and comparable to half angle slicing, and since no precomputation is used, no illumination updates need to be performed. Accordingly, also the memory footprint is low, as only the additional occlusion buffers need to be allocated. Finally, clipping planes can be used interactively.

Patel et al. apply concepts similar to directional occlusion shading when visualizing seismic volume data sets [PBVG10]. By using the incremental blurring operation modeled through the phase function, they are able to visualize seismic data sets without emphasizing the data inherent noise. A more recent extension by Schott et al. allows to integrate geometry and support light interactions between the geometry and the volumetric media [SMGB12].

Multidirectional Occlusion Shading introduced by Šoltészová et al. [vPBV10], extends the directional occlusion technique [SPH*09] to allow for a more flexible placement of the light source. Therefore, the convolution kernel is exchanged, which is used for the iterative convolution performed during the slicing. Directional occlusion shading makes the assumption, that the light source direction is aligned with the viewing direction, and therefore does not allow to change the light source position. Multidirectional occlusion shading avoids this assumption, by introducing more complex convolution kernels. Instead of performing the convolution of the opacity buffer with a symmetrical disc-shaped kernel, Šoltészová et al. propose the usage of an elliptical kernel derived from a tilted cone c. In their paper, they derive the convolution kernel from the light source position and the aperture θ defining c. Due to the directional component introduced by the front-to-back slicing, the tilt angle of the cone is limited to lie in $[0, \frac{\pi}{2} - \theta]$, which restricts the shape of the filter kernel from generating into hyperbolas or parabolas. The underlying illumination model is the same as with directional occlusion shading, whereas the indirect lighting contribution is modified by exploiting ellipse shaped filter kernels.

As standard directional occlusion shading, multidirectional occlusion shading uses front-to-back slice rendering, which tightly binds it to this rendering paradigm. Since the introduced filter kernels allow to model light sources at different positions, multiple light sources are supported. However, due to the directional nature, light sources must be positioned in the viewer's hemisphere. Shadowing and scattering capabilities are the same as with directional occlusion shading, i. e., soft shadows and first order scattering is supported. However, since different light positions are supported, shadows can be made more prominent in the final image when

the light direction differs from the view direction. This effect is shown in Figure 7, where Figure 7 *(middle)* shows a rendering where the light source is located in the camera position, while the light source has been moved to the left in Figure 7 *(right)*. Due to the more complex filter kernel, the rendering time of this model is slightly higher than when using directional occlusion shading. The memory consumption is equally low.

4.3. Light Space Based Techniques (■)

As light space based techniques we consider all those techniques, where illumination information is directionally propagated with respect to the current light position. Unlike the slice based techniques, these techniques are independent of the underlying rendering paradigm. However, since illumination information is propagated along a specific direction, these techniques usually support only a single light source.

Deep Shadow Mapping has been developed to enable shadowing of complex, potentially semi-transparent, structures [LV00]. While standard shadow mapping [Wil78] supports basic shadowing by projecting a shadow map as seen from the light source [SKvW*92], it does not support semi-transparent occluders which often occur in volume rendering. In order to address semi-transparent structures, opacity shadow maps serve as a stack of shadow maps, which store alpha values instead of depth values in each shadow map [KN01]. Nevertheless, deep shadow maps are a more compact representation for semi-transparent occluders. They also consist of a stack of textures, but in contrast to the opacity shadow maps, an approximation to the shadow function is stored in these textures. Thus, it is possible to approximate shadows by using fewer hardware resources. Deep shadow mapping has first been applied to volume rendering by Hadwiger et al. [HKSB06]. An alternative approach has been presented by Ropinski et al. [RKH08]. While the original deep shadow map approach [LV00] stores the overall light intensity in each layer, in volume rendering it is advantageous to store the absorption given by the accumulated alpha value in analogy to the volume rendering integral. Thus, for each shadow ray, the alpha function is analyzed, i. e., the function describing the absorption, and approximated by using linear functions.

To have a termination criterion for the approximation of the shadowing function, the depth interval covered by each layer can be restricted. However, when it is determined that the currently analyzed voxels cannot be approximated sufficiently by a linear function, smaller depth intervals are considered. Thus, the approximation works as follows. Initially, the first hit point for each shadow ray is computed, similar as done for standard shadow mapping. Next, the distance to the light source of the first hit point and the alpha value for this position are stored within the first layer of the deep shadow map. At the first hit position, the alpha value usually equals zero. Starting from this first hit point, each shadow ray is traversed and it is checked iteratively whether the samples encountered so far can be approximated by a linear function. When processing a sample, where this approximation would not be sufficient, i. e., a user defined error is exceeded, the distance of the previous sample to the light source as well as the accumulated alpha value at the previous sample are stored in the next layer of the deep shadow map. This is repeated until all layers of the deep shadow map have been created.

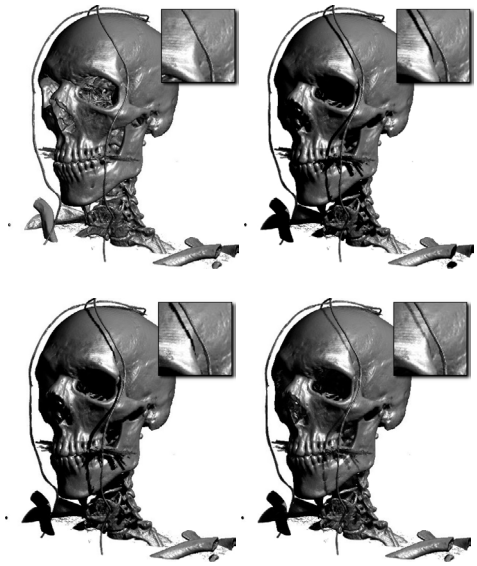

Figure 8: *The visible human head data set rendered without shadows (top left), with shadow rays (top right), with shadow mapping (bottom left) and with deep shadow maps (bottom right) (images taken from [RKH08]).*

The error threshold used when generating the deep shadow map data structure is introduced to determine whether the currently analyzed samples can be approximated by a linear function. In analogy to the original deep shadow mapping technique [LV00], this error value constrains the variance of the approximation. This can be done by adding (resp. subtracting) the error value at each sample's position. When the alpha function does not lie anymore within the range given by the error threshold, a new segment to be approximated by a linear function is started. A too small error threshold results in a too close approximation, and more layers are needed to represent the shadow function.

Once the deep shadow map data structure is obtained, the resulting shadowing information can be used in different ways. A common approach is to use the shadowing information in order to diminish the diffuse reflectance L_{diff} when using gradient-based shading. Nevertheless, it can also be used to directly modulate the emissive contribution L_e at $\vec{x}$.

A comparison of deep shadow mapping, which allows semi-transparent occluders, with shadow rays and standard shadow mapping is shown in Figure 8. As it can be seen deep shadow maps introduce artifacts when thin occluder structures are present. The shadows of these structures show transparency effects although the occluders are opaque. This results from the fact that an approximation of the alpha function is exploited and especially thin structures may diminish when approximating over too long depth intervals.

The deep shadow mapping technique is not bound to a specific rendering paradigm. It supports directional and point light sources, which should lie outside the volume since the shadow propagation is unidirectional. Due to the same reason, only a single light source is supported. Both, rendering time and update of the deep shadow data structure, which needs to be done when lighting or transfer function changes, are interactive. The memory footprint of deep shadow mapping is directly proportional to the number of used layers.

Desgranges et al. [DEP05] also propose a gradient-free technique for shading and shadowing which is based on projective texturing. Their approach produces visually pleasing shadowing effects and can be applied interactively.

Image Plane Sweep Volume Illumination [SYR11] allows the integration of advanced illumination effects into a GPU-based volume ray-caster by exploiting the plane sweep paradigm. By using sweeping, it becomes possible to reduce the illumination computation complexity and achieve interactive frame rates, while supporting scattering as well as shadowing. The approach is based on a reformulation of the optical model, that either considers only single scattering with a single light direction, or multiple scattering resulting from a rather high albedo [MC10]. To have shadowing effects of higher frequency and more diffuse scattering effects at the same time, these two formulations are combined. This is achieved by adding the local emission at $\vec{x}$ with the following scattering term:

$$g(\vec{x},\omega_o) = (1-a(\vec{x}))\cdot p(\vec{x},\omega_l,\omega_o)\cdot I_{ss}(\vec{x},\omega_l) + a(\vec{x})\cdot\int_\Omega p(\vec{x},\omega_i,\omega_o)\cdot I_{ms}(\vec{x},\omega_i)d\omega_i. \quad (25)$$

Here, ω_l is the principal light direction and ω_i are all directions from which light may enter. $a(\vec{x})$ is the albedo, which is the probability that light is scattered rather than absorbed. Thus, $I_{ss}(\vec{x},\omega_l)$ describes the single scattering contribution, and $I_{ms}(\vec{x},\omega_i)$ describes the multiple scattering contribution. Similar to the formulation presented by Max and Chen [MC10], the single scattering contribution is written as:

$$I_{ss}(\vec{x},\omega_l) = T(\vec{x}_l,\vec{x})\cdot L_0 + \int_{\vec{x}_l}^{\vec{x}} T(\vec{x}',\vec{x})\cdot\tau(\vec{x}')\cdot L_e(\vec{x}')d\vec{x}'. \quad (26)$$

Multiple scattering is simulated by assuming the presence of a high albedo making multiple scattering predominant and allowing to approximate it by using a diffusion approach:

$$I_{ms}(\vec{x},\omega_i) = \nabla_{diff}\int_{\vec{x}_b}^{\vec{x}} \frac{1}{|\vec{x}'-\vec{x}|}\cdot L_e(\vec{x}')d\vec{x}'. \quad (27)$$

Here, ∇_{diff} represents the diffusion approximation and $\frac{1}{|\vec{x}'-\vec{x}|}$ results in a weighting, such that more distant samples have less influence. $\vec{x}_b$ represents the background position just lying outside the volume. In practice the computation of the scattering contributions can be simplified, when assuming the presence of a forward peaked phase function, such that all relevant samples lie in direction of the light source. Thus, the computation can be performed in a synchronized manner, which is depicted by the following formulation for single scattering:

$$\begin{aligned} I'_{ss}(\vec{x},\omega_i) =& T(\vec{x}_l,\vec{x})\cdot L_0 \\ &+ \int_{\vec{x}''}^{\vec{x}} T(\vec{x}',\vec{x})\cdot\tau(\vec{x}')\cdot L_e(\vec{x}')d\vec{x}' \\ &+ \int_{\vec{x}_l}^{\vec{x}''} T(\vec{x}',\vec{x})\cdot\tau(\vec{x}')\cdot L_e(\vec{x}')d\vec{x}', \end{aligned} \quad (28)$$

and a similar representation for the multiple scattering contribution:

$$\begin{aligned} I'_{ms}(\vec{x},\omega_i) =& \nabla_{diff}\int_{\vec{x}''}^{\vec{x}} \frac{1}{|\vec{x}'-\vec{x}|}\cdot L_e(\vec{x}')d\vec{x}' + \\ & \nabla_{diff}\int_{\vec{x}_b}^{\vec{x}''} \frac{1}{|\vec{x}'-\vec{x}|}\cdot L_e(\vec{x}')d\vec{x}'. \end{aligned} \quad (29)$$

Image plane sweep volume illumination facilitates this splitting at $\vec{x}''$, which is based on the current state of the sweep line, by performing a synchronized ray marching. During this process, the view rays are processed in an order, such that those being closer to the light source in image space are processed earlier. This is ensured by exploiting a modified plane sweeping approach, which iteratively computes the influence of structures when moving further away from the light source. To store the illumination information accumulated along all light rays between the samples of a specific sweep plane, a 2D illumination cache in form of a texture is used.

Image plane sweep volume illumination has been developed as a ray-based technique and thus can only be used with this rendering paradigm. It supports one point or directional light source, and simulates global shadowing and multiple scattering effects. Since all illumination computations are performed directly within a single rendering pass, it does not require any preprocessing does not need to store intermediate results within an illumination volume, which results in a low memory footprint. The interactive application of clipping planes is inherently supported.

Shadow Splatting has been proposed first in 2001 by Nulkar and Mueller [NM01]. As the name implies, it is bound to the splatting rendering paradigm, which is exploited to attenuate the lighting information as it travels through the volume. Therefore, a shadow volume is introduced, which is update in a first splatting pass, during which light information is attenuated as it travels through the volume. In the second splatting pass, which is used for rendering, the shadow volume is fetched to acquire the incident illumination. The indirect light computation in the first pass can be described as:

$$L_i(\vec{x},\vec{\omega}_i) = L_l \cdot T(\vec{x}_l,\vec{x}). \tag{30}$$

The actual shading is performed using Phong shading, whereas the diffuse and the specular lighting intensity is replaced by the indirect light $L_i(\vec{x},\vec{\omega}_i)$.

Zhang and Crawfis [ZC02, ZC03] present an algorithm, which does not require to store a shadow volume in memory. Therefore, their algorithm exploits two additional image buffers for handling illumination information. These buffers are used to attenuate the light during rendering, and to add its contribution to the final image. Whenever a contribution is added to the final image buffer seen from the camera, this contribution is also added to the shadow buffer as seen from the light source.

Shadow splatting can be used for multiple point light and distant light sources, as well as textured area lights. While the initial algorithm [ZC02] only supports hard shadows, a more recent extension integrates soft shadows [ZC03]. When computing shadows, the rendering time is approximately doubled with respect to regular splatting. The algorithm by Zhang and Crawfis [ZC02, ZC03] updates the 2D image buffers during rendering and thus requires no considerable update times, while the algorithm by Nulkar and Mueller [NM01] needs to update the shadow volume, which requires extra memory and update times. With a more recent extension, the integration of geometry also becomes possible [ZXC05].

4.4. Lattice Based Techniques ( )

As lattice based techniques, we classify all those approaches, that compute the illumination directly based on the underlying lattice. While many of the previous approaches are computing the illumination per sample during rendering, the lattice based techniques perform the computations per voxel, usually by exploiting the nature of the lattice. In this subsection, we also focus on those techniques which allow interactive data exploration, though it should be mentioned that other techniques working on non-regular grids exist [QXF*07].

Shadow Volume Propagation is a lattice-based approach, where illumination information is propagated in a preprocessing through the regular grid defining the volumetric data set. The first algorithm realizing this principle has been proposed by Behrens and Ratering in the context of slice based volume rendering [BR98]. Their technique simulates shadows caused by attenuation of one distant light source. These shadows, which are stored within a shadow volume, are computed by attenuating illumination slice by slice through the volume. More recently, the lattice-based propagation concept has been exploited also by others. Ropinski et al. have exploited such a propagation to allow high frequency shadows and low frequency scattering simultaneously [RDRS10b]. Their approach has been proposed as a ray-casting-based technique that approximates the light propagation direction in order to allow efficient rendering. It facilitates a four channel illumination volume to store luminance and scattering information obtained from the volumetric dataset with respect to the currently set transfer function. Therefore, similar to Kniss et al. [KPH*03], indirect lighting is incorporated by blurring the incoming light within a given cone centered about the incoming light direction. However, different to [KPH*03] shadow volume propagation uses blurring only for the chromaticity and not the intensity of the light (luminance). Although this procedure is not physically correct, it helps to accomplish the goal of obtaining harder shadow borders, which according to Wanger improve the perception of spatial structures [WFG92, Wan92]. Thus, the underlying illumination model can be specified as follows:

$$\begin{aligned} L(\vec{x},\vec{\omega}_o) =& L_0(\vec{x}_0,\vec{\omega}_o)\cdot T(\vec{x}_0,\vec{x}) \\ &+ \int_{\vec{x}_0}^{\vec{x}} (L_e(\vec{x}) + s(\vec{x},\vec{\omega}_i,\vec{\omega}_o))\cdot T(\vec{x}',\vec{x})d\vec{x}', \end{aligned} \tag{31}$$

where $s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ is calculated as the transport color $t(\vec{x})$, as assigned through the transfer function, multiplied by the chromaticity $c(\vec{x},\vec{\omega}_o)$ of the in-scattered light and modulated by the attenuated luminance $l_i(\vec{x},\vec{\omega}_i)$:

$$s(\vec{x},\vec{\omega}_i,\vec{\omega}_o) = t(\vec{x})\cdot c(\vec{x},\vec{\omega}_o)\cdot l_i(\vec{x},\vec{\omega}_i). \tag{32}$$

The transport color $t(\vec{x})$ as well as the chromaticity $c(\vec{x},\vec{\omega}_o)$ are wave length dependent, while the scalar $l_i(\vec{x},\vec{\omega}_i)$ describes the incident (achromatic) luminance. Multiplying

all together results in the incident radiance. Chromaticity $c(\vec{x},\vec{\omega}_o)$ is computed from the in-scattered light

$$c(\vec{x},\vec{\omega}_o) = \int_\Omega \tau(\vec{x}) \cdot p(\vec{x},\vec{\omega}_i{}',\vec{\omega}_o) \cdot c(\vec{x},\vec{\omega}_i{}')d\vec{\omega}_i{}', \qquad (33)$$

where Ω is the unit sphere centered around $\vec{x}$. To allow the unidirectional illumination propagation, $p(\vec{x},\vec{\omega}_i{}',\vec{\omega}_o)$ has been chosen to be a strongly forward peaked phase function:

$$p(\vec{x},\vec{\omega}_i{}',\vec{\omega}_o) = \begin{cases} (\vec{\omega}_i{}' \cdot \vec{\omega}_o)^\beta & \text{if } \vec{\omega}_i{}' \cdot \vec{\omega}_o < \theta(\vec{x}) \\ 0 & \text{otherwise.} \end{cases} \qquad (34)$$

The cone angle θ is used to control the amount of scattering and depends on the intensity at position $\vec{x}$, and the phase function is a Phong lobe whose extent is controlled by the exponent β, restricted to the cone angle θ.

During rendering, the illumination volume is accessed to look up the luminance value and scattering color of the current voxel. The chromaticity $c(\vec{x},\vec{\omega}_o)$ and the luminance $l_i(\vec{x},\vec{\omega}_i)$ are fetched from the illumination volume and used within $s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$ as shown above. In cases, where a high gradient magnitude is present, specular reflections and thus surface-based illumination is also integrated into $s(\vec{x},\vec{\omega}_i,\vec{\omega}_o)$.

Shadow volume propagation can be combined with various rendering paradigms, and does not require a noticeable amount of preprocessing, since the light propagation is carried out on the GPU. It supports point and directional light sources, whereby a more recent extension also supports area light sources [RDRS10a]. While the position of the light source is unconstrained, the number of light sources is limited to one, since only one illumination propagation direction is supported. By simulating the diffusion process, shadow volume propagation supports multiple scattering. Rendering times are quite low, since only one additional 3D texture fetch is required to obtain the illumination values. However, when the transfer function is changed, the illumination information needs to be updated and thus the propagation step needs to be recomputed. Nevertheless, this still supports interactive frame rates. Since the illumination volume is stored and processed on the GPU, a sufficient amount of graphics memory is required.

Piecewise Integration [HLY08] computes global light transport by dividing rays into k segments and evaluating the incoming radiance using:

$$L_i(\vec{x},\vec{\omega}_i) = L_0 \cdot \prod_{n=0}^{k} T(\vec{x}_n,\vec{x}_{n+1}). \qquad (35)$$

Short rays are first cast towards the light source for each voxel and the piecewiese segments are stored in an intermediate 3D texture using a multiresolution datastructure [LLY06]. Then, again for each voxel, global rays are sent towards the light source now taking steps of size equal to the short rays and sampling from the piecewise segment 3D texture. In addition to direct illumination, first order inscattering is approximated by treating scattering in the vicinity as emission. To preserve interactivity, the final radiance is progressively refined by sending one ray at a time and blending the result into a final 3D texture.

The piecewise integration technique can be combined with other rendering paradigms, even though it has been developed for volume ray-casting. It supports a single directional or point light source, which can be dynamically moved, and it include an approximation of first order inscattering. Rendering times are low, since only a single extra lookup is required to compute the incoming radiance at the sample position. As soon at the light source or transfer function changes, the radiance must be recomputed which can be performed interactively and progressively. Three additional 3D textures are required for the computations, however a multiresolution structure and lower resolution grid are used to alleviate some of the additional memory requirements.

Summed Area Table techniques were first exploited by Diaz et al. [DYV08] for computation of various illumination effects in interactive direct volume rendering. The 2D summed area table (SAT) is a data structure where each cell (x,y) stores the sum of every cell located between the initial position $(0,0)$ and (x,y), such that the 2D SAT for each pixel p can be computed as:

$$SAT_{image}(x,y) = \sum_{i=0,j=0}^{x,y} p_{i,j} \qquad (36)$$

A 2D SAT can be computed incrementally during a single pass and the computation cost increase linearly with the number of cells. Once a 2D SAT has been constructed, an efficient evaluation of a region can be performed with only four texture accesses to the corners of the regions to be evaluated. Diaz et al. [DYV08] created a method, known as Vicinity Occlusion Mapping (VOM), which effectively utilizes two SATs computed from the depth map, one which stores the accumulated depth for each pixel and one which stores the number of contributed values of the neighborhood of each pixel. The vicinity occlusion map is then used to incorporate view dependent ambient occlusion and halo effects in the volume rendered image. The condition for which a pixel should be occluded is determined by evaluating the average depth around that pixel. If this depth is larger than the depth of the pixel, the neighboring structures are further away which means this structure is not occluded. If the structure in the current pixel is occluded, the amount of occlusion, i. e., how much the pixel should be darkened is determined by the amount of differences of depths around the

corresponding pixel. A halo effect can be achieved by also considering the average depth from the neighborhood. If the structure in the pixel lies outside the object, a halo color is rendered which decays with respect to the distance to the object.

Density Summed Area Table [DVND10] utilize a 3D SAT for ambient occlusion, calculated from the density values stored in the volume. This procedure is view independent and supports incorporation of semi-transparent structures, while it is more accurate than the 2D image space approach [DYV08]. However, while only 8 texture accesses are needed to estimate a region in a 3D SAT, the preprocessing is more computationally expensive and it results in a larger memory footprint. However, the 3D SAT does not need to be re-computed if the view changes. Desgranges and Engel also describe a SAT based inexpensive approximation of ambient light [DE07]. They combine ambient occlusion volumes from different filtered volumes into a composite occlusion volume.

Extinction-based Shading and Illumination is recent technique by Schlegel et al. [SMP11], where a 3D SAT is used to model ambient occlusion, color bleeding, directional soft shadows and scattering effects by using the summation of the exponential extinction coefficient. As the summation of the exponential extinction coefficient in their model is order independent, the use of a 3D SAT and efficient aggregate summation decreases the computational cost significantly. The directional soft shadows and scattering effects are incorporated by estimating a cone defined by the light source. The approximation of that cone is done with a series of cuboids. Texture accesses in the SAT structure can only be performed axis-aligned, thus the cuboids are placed along the two axis-aligned planes with the largest projection size of the cone towards the light. Schlegel et al. do this by defining the primary cone axis to be the axis with the smallest angle to the vector towards the light source and then defining two secondary axes (of four possibilities) which are afterwards used to form two axis-aligned planes. A selections of renderings using this method is shown in Figure 9.

Extinction-based Shading and Illumination is not limited to any specific rendering paradigm. The technique can handle point, directional and area light sources. For each additional light source, the extiction must be estimated which means that rendering time increases with the number of light sources. In terms of memory consumption an extra 3D texture is required for the SAT which can be of lower resolution than the volume data while still producing convincing results.

4.5. Basis Function Based Techniques ( )

The group of basis function based techniques covers those approaches, where the light source radiance L_l and transparency (visibility) $T(\vec{x}_l,\vec{x})$ from the position $\vec{x}$ towards the

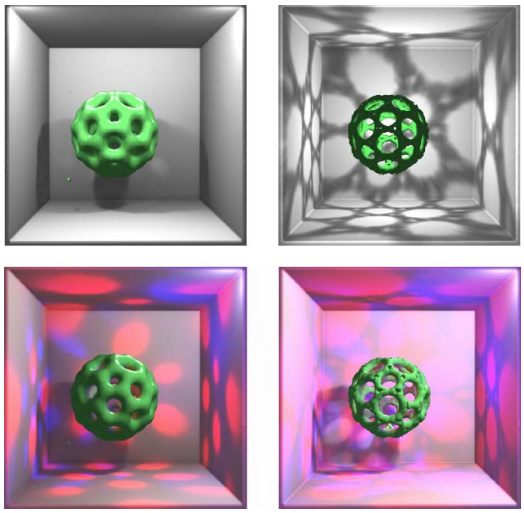

Figure 9: *Extinction-based Shading and Illumination [SMP11] allows interactive integration of local and global illumination properties and advanced light setups. (Images taken from [SMP11], courtesy of Philipp Schlegel)*

light source position $\vec{x}_l$ are represented using a basis function to compute the incident radiance in equation 3. Since radiance and visibility are computed independently, light sources may be dynamically changed without an expensive visibility update. Furthermore, the light source composition may be complex as they are projected to another basis which is independent of the number of light sources. Up to now, the only basis functions used in the area of volume rendering are spherical harmonics. However, to be able to incorporate future techniques into our classification, we have decided to keep this group more general. There are two key properties of spherical harmonics which are important and thus worth special mentioning. The first is, that efficient integral evaluation can be performed, which supports real-time computation of the incoming radiance over the sphere. The second property is, that it can be rotated on the fly, and thus supports dynamic light sources. Since it is beyond the scope of this article to provide a detailed introduction to spherical harmonic based lighting, we focus on the volume rendering specific properties only, and refer to the work by Sloan et al. [SKS02] for further details. While spherical harmonics were first used in volume rendering by Beason et al. [BGB*06] to enhance isosurface shading, in this section, we cover algorithms applying spherical harmonics in direct volume rendering.

Spherical Harmonics have been first applied to direct volume rendering by Ritschel [Rit07]. He has applied spherical basis functions to simulate low-frequency shadowing effects. To enable interactive rendering, a GPU-based approach is exploited for the computation of the required spherical harmonic coefficients. Therefore, a multiresolution data structure similar to a volumetric mipmap is computed

and stored on the GPU. Whenever the spherical harmonic coefficients need to be recomputed, this data structure is accessed and the ray traversal is performed with increasing step size. The increasing step size is realized by sampling the multiresolution data structure in different levels, whereby the level of detail decreases as the sampling proceeds further away from the voxel for which the coefficients need to be computed.

In a more recent paper spherical harmonics were used to support more advanced material properties [LR10]. Therefore, the authors exploit the approach proposed by Ritschel [Rit07] to compute spherical harmonic coefficients. During ray-casting, the spherical integral over the product of an area light source and voxel occlusion is efficiently computed using the precomputed occlusion information. Furthermore, realistic light material interaction effects are integrated while still achieving interactive volume frame rates. By using a modified spherical harmonic projection approach, tailored specifically for volume rendering, it becomes possible to realize non cone-shaped phase functions in the area of interactive volume rendering. Thus, more realistic material representations become possible.

One problem with using a basis function such as spherical harmonics is the increased storage requirement. Higher frequencies can be represented when using more coefficient, thus obtaining better results. However, each coefficient needs to be stored to be used during rendering. Kronander et al. [KJL*12] address this storage requirement issue and also decrease the time it takes to update the visibility. In their method, both volume data and spherical harmonic coefficients exploit the multiresolution technique of Ljung [LLY06]. The sparse representation of volume data and visibility allows substantial reductions with respect to memory requirements. To also decrease the visibility update time, a two pass approach is used. First, local visibility is computed over a small neighborhood by sending rays uniformly distributed on the sphere. Then, piecewise integration of the local visibility is performed in order to compute global visibility [HLY08]. Since both, local and global visibility have been computed, the user can switch between them during rendering and thus reveal structures hidden by global features while still keeping spatial comprehension.

The spherical harmonics based methods discussed here are not bound to any specific rendering paradigm, although only volume ray-casting has been used by the techniques presented here. An arbitrary number of dynamic directional light sources are supported, i. e. environment maps. Area and point light sources are also supported, each adding an overhead since the direction towards the sample along the ray changes and the spherical harmonics representation of the light source must be rotated. Very naturally looking images can be generated due to the possibility of complex light setups and material representation (see Figure 10), in contrast to the hard shadows of techniques like half angle slicing.

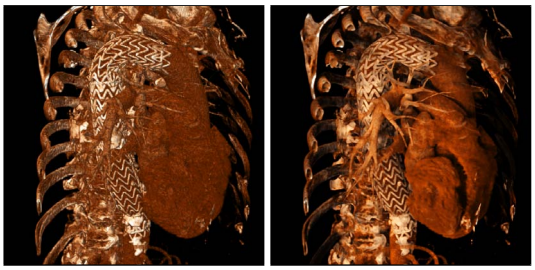

Figure 10: *A compute tomography scan of a human torso rendered with diffuse gradient-based shading (left) and spherical harmonic lighting (right). (Images taken from [KJL*12])*

However, storage and computation time limits the accuracy of the spherical harmonics representation thus restricting the illumination to low frequency effects. In general, light sources are constrained to be outside of the volume, however when using local visibility, e. g., as in [KJL*12], they may be positioned arbitrarily. The rendering time is interactive, only requiring some (in general 1-4) extra texture lookups and scalar products. The light source representation must be recomputed or rotated when the illumination changes but this is generally fast and can be performed in real-time. However, the visibility must be recomputed when the transfer function or clip plane changes, which is expensive but can be performed interactively [KJL*12]. The memory requirements are high as each coefficient needs to be stored for the visibility in the volume. We refer to the work by Kronander et al. [KJL*12] for an indepth analysis of how to trade off memory size and image quality.

4.6. Ray-Tracing Based Techniques ( )

The work discussed above focuses mainly on increasing the lighting realism in interactive volume rendering applications. This is often achieved by making assumptions regarding the lighting setup or the illumination propagation, e. g., assuming that only directional scattering occurs. As ray-tracing has been widely used in polygonal rendering to support physically correct and thus realistic images, a few authors have also investigated this direction in the area of volume rendering. Gradients, as used for the gradient-based shading approaches, can also be exploited to realize a volumetric ray-tracer supporting more physically correct illumination. One such approach, considering only first order rays to be traversed, has been proposed by Stegmaier et al. [SSKE05]. With their approach they are able to simulate mirror-like reflections and refraction, by casting a ray and computing its deflection. To simulate mirror-like reflections, they perform an environment texture lookup. More sophisticated refraction approaches have been presented by others [LM05, RC06].

To produce images of high realism, stochastic Monte Carlo ray-tracing methods have also been applied in the area of interactive volume rendering. The benefit of these approaches is, that they are directly coupled with physically based light transport. The first Monte Carlo based approach has been presented by Rezk-Salama [RS07]. It supports shadowing and scattering, whereby interactivity is achieved by constraining the expensive scattering computations to selected surfaces only. With respect to the gradient magnitude it is decided, if a boundary surface, potentially initiating a scattering event, is present. The approach supports interactive frame rates and rendering of images with a high degree of realism.

More recently, Kroes et al. have presented an interactive Monte Carlo ray-tracer, which does not limit the scattering to selected boundary surfaces [KPB11]. Their approach simulates several real-world factors influencing volumetric illumination, such as multiple arbitrarily shaped and textured area light sources, a real-world camera with lens and aperture, as well as complex material properties. This combination allows to generate images showing a high degree of realism (see Figure 11). To adequatly represent both, volumetric and surface-like materials, their approach blends between BRDF based and a phase function based material representation.

4.7. Isosurface Based Techniques

Besides the algorithms discussed above, which can be combined with direct volume rendering and are therefore the main focus of this article, several approaches exist that allow advanced illumination when visualizing isosurfaces. While when dealing with direct volume rendering, a transfer function may extract several structures having different intensities and thus optical properties, when dealing with the simplest form of isosurface rendering, only a single isosurface with homogeneous optical properties is visible. This reduces the combinations of materials which interact with each other and produce complex lighting effects drastically. Several approaches have been proposed, which exploit this reduced complexity and thus allow advanced illumination with higher rendering speeds and sometimes lower memory footprints. Vicinity Shading [Ste03], which simulates illumination of isosurfaces by taking into account neighboring voxels, was the first approach addressing advanced illumination for isosurfaces extracted from volume data. In a precomputation the vicinity of each voxel is analyzed and the resulting value, which represents the occlusion of the voxel, is stored in a shading texture which can be accessed during rendering. Wyman et al. [WPSH06] and Beason et al. [BGB*06] focus on a spherical harmonic based precomputation, supporting the display of isosurfaces under static illumination conditions. Penner et al. exploit an ambient occlusion approach to render smoothly shaded isosurfaces [PM08]. More recently, Banks and Beacon have demonstrated how to decouple illumination sampling from isosurface generation, in order to achieve sophisticated isosurface illumination results [BB09].

Figure 11: *Monte Carlo ray-tracing allows to incorporate gradient-based material properties and the modeling of camera apertures to generate highly realistic images. (Image taken from [KPB11], courtesy of Thomas Kroes)*

5. Perceptual Impact

Besides the technical insights and the achievable illumination effects described above, the perceptual impact of the current techniques is also of relevance. Although, the perceptual qualities of advanced illumination techniques are frequently pointed out, remarkably little research has been done in order to investigate them in a systematic fashion. One study showing the perceptual benefits of advanced volumetric illumination models has been conducted with the goal to analyze the impact of the shadow volume propagation technique [RDRS10b]. The users participating in this study had to perform several tasks depending on the depth perception of static images, where one set was generated using gradient-based shading and the other one using shadow volume propagation. The results of the study indicate, that depth perception is more accurate and performed faster when using shadow volume propagation.

Šoltészová et al. proposed and evaluated the perceptual impact of chromatic soft shadows in the context of interactive volume rendering [vPV11]. Inspired by illustrators, they replace the luminance decrease usually present in shadowed areas by a chromaticity blending. In their user study, they could show the perceptual benefit of these chromatic shadows with respect to depth and surface perception. However, their results also indicate that brighter shadows might have a negative impact on the perceptual qualities of an image, and should therefore be used with consideration.

More recently, Lindemann and Ropinski presented the results of a user study, in which they have investigated the perceptual impact of seven volumetric illumination techniques [LR11]. Within their study, depth and volume related tasks had to be performed. They could show, that advanced volumetric illumination models make a difference when it is necessary to assess depth or size in images. However, they could not find a relation between the time used to perform a certain task and the used illumination model. The results

of this study indicate, that the directional occlusion shading model has the best perceptual capabilities [LR11].

6. Usage Guidelines

To support application developers when choosing an advanced volumetric shading method, we provide a comparative overview of some capabilities within this section. Table 1 contains an overview of the properties discussed along with each algorithm description. However, in some cases more detailed information is necessary to compare and select techniques for specific application cases. Therefore, we provide the comparative charts shown in Figure 12.

Figure 12 *(a)* provides a comparative overview of the techniques belonging to the discussed groups with respect to illumination complexity and underlying rendering paradigm. To be able to assess the illumination capabilities of the incorporated techniques, the illumination complexity has been depicted as a continuous scale along the x-axis. It ranges from low complexity lighting, e. g., local shading with a single point light source, to highly complex lighting, which could for instance simulate scattering and shadowing as obtained from multiple area light sources. The y-axis in Figure 12 depicts on the other hand which algorithmic rendering paradigm is supported. Thus, Figure 12 *(a)* depicts for each technique with which rendering paradigm it has been used and what lighting complexity has been achieved. We provide this chart as a reference, since we believe that the shown capabilities are important criteria to be assessed when choosing an advanced volumetric illumination model.

Figure 12 *(b)* depicts the scalability of the covered techniques in terms of growing image and data set size with respect to performance. Since the x-axis depicts scalability with respect to data size and the y-axis depicts scalability with respect to image size, the top right quadrant for instance contains all algorithms which allow a good performance in this respect. When selecting techniques in Figure 12 *(a)* based on the illumination complexity and the underlying rendering paradigm, Figure 12 *(b)* can be used to compare them regarding their scalability.

7. Future Challenges

Though the techniques presented in this article allow to generate realistic volume rendered images at interactive frame rates, still several challenges exist to be addressed in future work. When reviewing existing techniques, it can be noticed that although the underlying optical models already contain simplifications, there is sometimes a gap between the model and the actual implementation. In the future it is necessary to narrow this gap by providing physically more accurate illumination models. The work done by Kroes et al. [KPB11] can be considered as a first valuable step into this direction. One important ingredient necessary to provide further realism is the support of advanced material properties. Current techniques only support simplified phase function models and sometimes BRDFs, while more realistic volumetric material functions are not yet integrated.

Another challenging area which needs to be addressed in the future, is the integration of several data sources. Although, Nguyen et al. [NEO*10] have presented an approach for integrating MRI and fMRI data, more general approaches potentially supporting multiple modalities are missing. Along this lines, also the integration of volumetric and geometric information needs more consideration. While the approach by Schott et al. [SMGB12] allows an integration in the context of slice based rendering, other volume rendering paradigms are not yet supported.

Apart from that, large data sets still pose a driving challenge, since the illumination computation takes a lot of time and due to memory limitations it is not possible to store large intermediate data sets.

8. Conclusions

Within this article, we have discussed the current state of the art regarding volumetric illumination models for interactive volume rendering. We have classified and explained the covered techniques and related their visual capabilities by using a uniform mathematical notation. Thus, the article can be used as both, a reference for the existing techniques, but also for deciding which advanced volumetric illumination approaches should be used in specific cases. We hope, that this helps researchers as well as application developers to make the right decisions early on and identify challenges for future work.

Acknowledgments

Many of the presented techniques have been integrated into the Voreen volume rendering engine (`www.voreen.org`), which is an open source visualization framework. The authors would like to thank Florian Lindemann, for implementing the half angle slicing used to generate Figure 1 *(right)*.

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. 17

[BGB*06] BEASON K. M., GRANT J., BANKS D. C., FUTCH B., HUSSAINI M. Y.: Pre-computed illumination for isosurfaces. In *Conference on Visualization and Data Analysis* (2006), pp. 1–11. 15, 17

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. *Proceedings of SIGGRAPH '77 11* (1977), 192–198. 6

[BR98] BEHRENS U., RATERING R.: Adding shadows to a texture-based volume renderer. In *IEEE Int. Symp. on Volume Visualization* (1998), pp. 39–46. 4, 13, 22

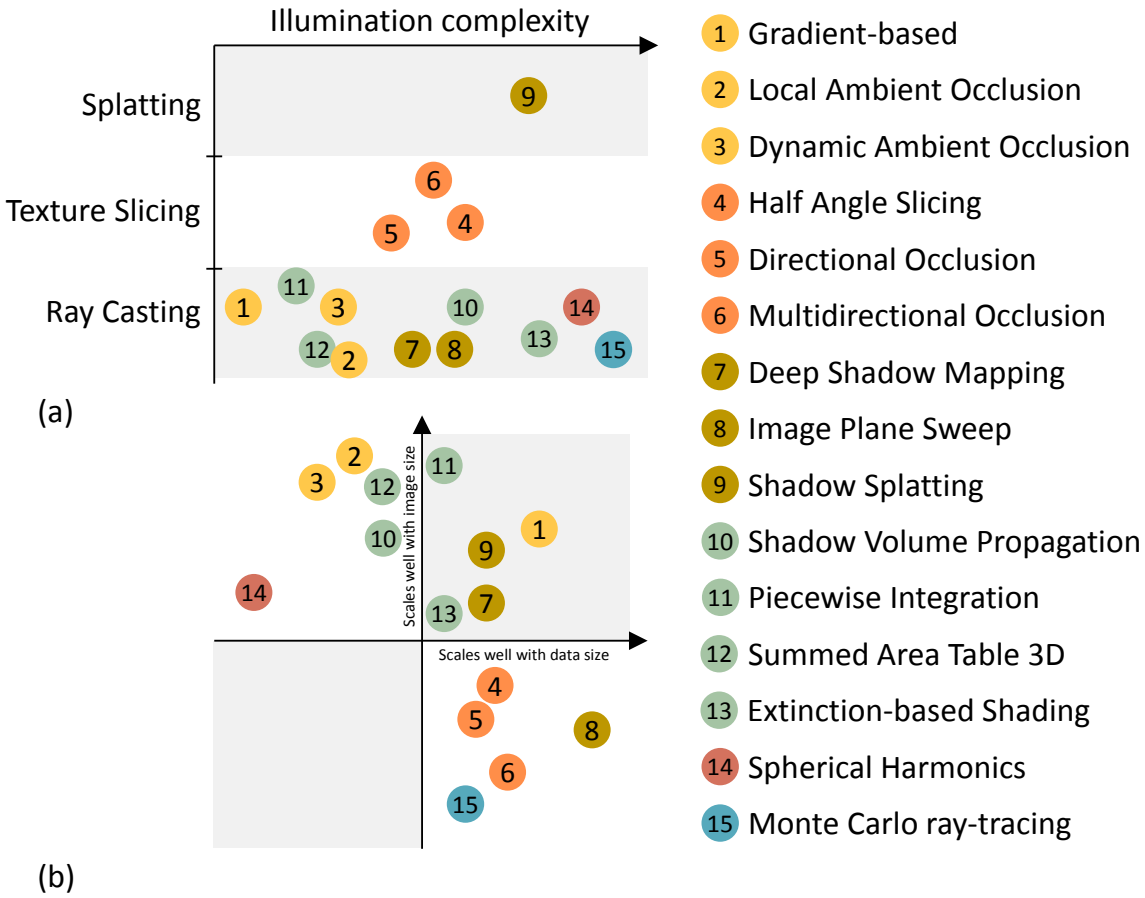

Figure 12: *In (a) the illumination complexity of a method is shown in relation to the underlying rendering paradigm, whereby the illumination complexity increases from left to right. Techniques to the far left can for instance only handle point lights, while those to the far right can handle area light sources as well as multiple scattering. Plot (b) shows how each method scales in terms of performance with respect to large image and data size. Techniques in the upper right quadrant perform well independent of image and data size while methods in the lower left quadrant degrade substantially for large screen and data sizes.*

[CCF94] CABRAL B., CAM N., FORAN J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 Symposium on Volume Visualization* (1994), pp. 91–98. 4, 9

[DE07] DESGRANGES P., ENGEL K.: US patent application 2007/0013696 A1: Fast ambient occlusion for direct volume rendering, 2007. 15

[DEP05] DESGRANGES P., ENGEL K., PALADINI G.: Gradient-free shading: A new method for realistic interactive volume rendering. In *Int. Fall Workshop on Vision, Modeling, and Visualization* (2005), pp. 209–216. 4, 12

[DK09] DACHSBACHER C., KAUTZ J.: Real-time global illumination. In *ACM SIGGRAPH Courses Program* (2009). 1

[DVND10] DÍAZ J., VÁZQUEZ P.-P., NAVAZO I., DUGUET F.: Real-time ambient occlusion and halos with summed area tables. *Computers and Graphics 34* (2010), 337–350. 15, 22

[DYV08] DÍAZ J., YELA H., VÁZQUEZ P.-P.: Vicinity occlusion maps - enhanced depth perception of volumetric models. In *Computer Graphics Int.* (2008), pp. 56–63. 14, 15

[GP06] GRIBBLE C. P., PARKER S. G.: Enhancing interactive particle visualization with advanced shading models. In *Proceedings of the 3rd symposium on Applied perception in graphics and visualization* (2006), pp. 111–118. 1

[HAM11] HOSSAIN Z., ALIM U. R., MÖLLER T.: Toward high-quality gradient estimation on regular lattices. *IEEE Transactions on Visualization and Computer Graphics 17* (2011), 426–439. 7

[HKSB06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K.: Gpu-accelerated deep shadow maps for direct volume render-

ing. In *ACM SIGGRAPH/EG Conference on Graphics Hardware* (2006), pp. 27–28. 11, 22

[HLY07] HERNELL F., LJUNG P., YNNERMAN A.: Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *IEEE/EG Int. Symp. on Volume Graphics* (2007), pp. 1–8. 7, 8

[HLY08] HERNELL F., LJUNG P., YNNERMAN A.: Interactive global light propagation in direct volume rendering using local piecewise integration. In *IEEE/EG Int. Symp. on Volume and Point-Based Graphics* (2008). 14, 16, 22

[HLY09] HERNELL F., LJUNG P., YNNERMAN A.: Local ambient occlusion in direct volume rendering. *IEEE Transations on Visualization and Computer Graphics 15*, 2 (2009), 548–559. 7, 8, 22

[KJL*12] KRONANDER J., JÖNSSON D., LÖW J., LJUNG P., YNNERMAN A., UNGER J.: Efficient visibility encoding for dynamic illumination in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 447–462. 5, 16, 22

[KKH02] KNISS J., KINDLMANN G., HANSEN C.: Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics 8* (2002), 270–285. 9

[KN01] KIM T.-Y., NEUMANN U.: Opacity shadow maps. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (2001), pp. 177–182. 11

[KPB11] KROES T., POST F. H., BOTHA C. P.: *Interactive Direct Volume Rendering with Physically-Based Lighting*. Preprint 2011-11, Delft University of Technology, 2011. 17, 18, 22

[KPH*03] KNISS J., PREMOZE S., HANSEN C., SHIRLEY P., MCPHERSON A.: A model for volume lighting and modeling. *IEEE Transations on Visualization and Computer Graphics 9*, 2 (2003), 150–162. 9, 10, 13, 22

[KPHE02] KNISS J., PREMOZE S., HANSEN C., EBERT D.: Interactive translucent volume rendering and procedural modeling. In *IEEE Visualization 2002* (2002), pp. 109–116. 2, 4, 9, 10

[KW03] KRÜGER J., WESTERMANN R.: Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Visualization '03* (2003). 4

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8* (1988), 29–37. 5, 6, 22

[LL94] LACROUTE P., LEVOY M.: Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), pp. 451–458. 4

[LLY06] LJUNG P., LUNDSTRÖM C., YNNERMAN A.: Multiresolution interblock interpolation in direct volume rendering. In *IEEE/EG Symposium on Visualization* (2006), p. 259Ű266. 7, 14, 16

[LM05] LI S., MUELLER K.: Accelerated, high-quality refraction computations for volume graphics. In *Volume Graphics, 2005. Fourth International Workshop on* (2005), pp. 73–229. 16

[LR10] LINDEMANN F., ROPINSKI T.: Advanced light material interaction for direct volume rendering. In *IEEE/EG Int. Symp. on Volume Graphics* (2010), pp. 101–108. 16, 22

[LR11] LINDEMANN F., ROPINSKI T.: About the influence of illumination models on image comprehension in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1922–1931. 1, 17, 18

[LV00] LOKOVIC T., VEACH E.: Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 385–392. 11

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transations on Visualization and Computer Graphics 1*, 2 (1995), 99–108. 1, 2, 3, 4, 5

[MC10] MAX N., CHEN M.: Local and global illumination in the volume rendering integral. In *Scientific Visualization: Advanced Concepts* (2010), pp. 259–274. 2, 12

[MMC99] MUELLER K., MÖLLER T., CRAWFIS R.: Splatting without the blur. In *IEEE Visualization* (1999), p. 61. 4

[MR10] MESS C., ROPINSKI T.: Efficient acquisition and clustering of local histograms for representing voxel neighborhoods. In *IEEE/EG Volume Graphics* (2010), pp. 117–124. 9

[NEO*10] NGUYEN T. K., EKLUND A., OHLSSON H., HERNELL F., LJUNG P., FORSELL C., ANDERSSON M. T., KNUTSSON H., YNNERMAN A.: Concurrent volume visualization of real-time fmri. In *IEEE/EG Volume Graphics* (2010), pp. 53–60. 7, 18

[NM01] NULKAR M., MUELLER K.: Splatting with shadows. In *Volume Graphics* (2001). 13

[PBVG10] PATEL D., BRUCKNER S., VIOLA I., GRÖLLER M. E.: Seismic volume visualization for horizon extraction. In *Proceedings of the IEEE Pacific Visualization Symposium 2010* (2010), pp. 73–80. 7, 10

[PM08] PENNER E., MITCHELL R.: Isosurface ambient occlusion and soft shadows with filterable occlusion maps. In *IEEE/EG Int. Symp. on Volume and Point-Based Graphics* (2008), pp. 57–64. 17

[QXF*07] QIU F., XU F., FAN Z., NEOPHYTOS N., KAUFMAN A., MUELLER K.: Lattice-based volumetric global illumination. *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 1576–1583. 13

[RBV*08] RUIZ M., BOADA I., VIOLA I., BRUCKNER S., FEIXAS M., SBERT M.: Obscurance-based volume rendering framework. In *IEEE/EG Int. Symp. on Volume and Point-Based Graphics* (2008), pp. 113–120. 8

[RC06] RODGMAN D., CHEN M.: Refraction in volume graphics. *Graph. Models 68* (2006), 432–450. 16

[RDRS10a] ROPINSKI T., DÖRING C., REZK SALAMA C.: Advanced volume illumination with unconstrained light source positioning. *IEEE Computer Graphics and Applications 30*, 6 (2010), 29–41. 14, 22

[RDRS10b] ROPINSKI T., DÖRING C., REZK-SALAMA C.: Interactive volumetric lighting simulating scattering and shadowing. In *IEEE Pacific Visualization* (2010), pp. 169–176. 4, 5, 13, 17, 22

[Rit07] RITSCHEL T.: Fast gpu-based visibility computation for natural illumination of volume data sets. In *Eurographics Short Paper Proceedings 2007* (2007), pp. 17–20. 15, 16, 22

[RKH08] ROPINSKI T., KASTEN J., HINRICHS K. H.: Efficient shadows for gpu-based volume raycasting. In *Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2008), pp. 17–24. 11

[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. 6, 8, 9, 22

[RS07] REZK-SALAMA C.: GPU-based monte-carlo volume raycasting. In *Pacific Graphics* (2007). 17

[RSHRL09] REZK SALAMA C., HADWIGER M., ROPINSKI T., LJUNG P.: Advanced illumination techniques for gpu volume raycasting. In *ACM SIGGRAPH Courses Program* (2009). 1

[SHC*09] SMELYANSKIY M., HOLMES D., CHHUGANI J., LARSON A., CARMEAN D. M., HANSON D., DUBEY P., AUGUSTINE K., KIM D., KYKER A., LEE V. W., NGUYEN A. D., SEILER L., ROBB R.: Mapping high-fidelity volume rendering for medical imaging to cpu, gpu and many-core architectures. *IEEE Transactions on Visualization and Computer Graphics 15* (2009), 1563–1570. 1, 4

[SKS02] SLOAN P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of SIGGRAPH '02* (2002), pp. 527–536. 15

[SKvW*92] SEGAL M., KOROBKIN C., VAN WIDENFELT R., FORAN J., HAEBERLI P.: Fast shadows and lighting effects using texture mapping. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992), Proceedings of SIGGRAPH '92, pp. 249–252. 11

[SMGB12] SCHOTT M., MARTIN T., GROSSET A., BROWNLEE C.: Combined surface and volumetric occlusion shading. In *Proceedings of Pacific Vis 2012* (2012). 10, 18

[SMP11] SCHLEGEL P., MAKHINYA M., PAJAROLA R.: Extinction-based shading and illumination in GPU volume raycasting. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1795–1802. 4, 15, 22

[SPH*09] SCHOTT M., PEGORARO V., HANSEN C., BOULANGER K., BOUATOUCH K.: A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum (Proceedings of Eurographics/IEEE VGTC Symposium on Visualization 2009) 28*, 3 (2009), 855–862. 4, 5, 9, 10, 22

[SSKE05] STEGMAIER S., STRENGERT M., KLEIN T., ERTL T.: A simple and flexible volume rendering framework for graphics-hardware-based raycasting. *Fourth International Workshop on Volume Graphics 2005* (2005), 187–241. 16

[Ste03] STEWART A. J.: Vicinity shading for enhanced perception of volumetric data. In *IEEE Visualization* (2003), pp. 355–362. 17

[SYR11] SUNDÉN E., YNNERMAN A., ROPINSKI T.: Image plane sweep volume illumination. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2125–2134. 4, 5, 12, 22

[vPBV10] ŠOLTÉSZOVÁ V., PATEL D., BRUCKNER S., VIOLA I.: A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum (Eurographics/IEEE VGTC Symp. on Visualization 2010) 29*, 3 (2010), 883–891. 4, 10, 22

[vPV11] ŠOLTÉSZOVÁ V., PATEL D., VIOLA I.: Chromatic shadows for improved perception. In *Proceedings of Non-Photorealistic Animation and Rendering (NPAR)* (2011), pp. 105–115. 17

[Wan92] WANGER L. C.: The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *SI3D '92: 1992 Symp. on Interactive 3D graphics* (1992), pp. 39–42. 1, 13

[WFG92] WANGER L. C., FERWERDA J. A., GREENBERG D. P.: Perceiving spatial relationships in computer-generated images. *IEEE Comput. Graph. Appl. 12*, 3 (1992), 44–51, 54–58. 1, 13

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH '78* (1978), pp. 270–274. 11

[WPSH06] WYMAN C., PARKER S., SHIRLEY P., HANSEN C.: Interactive display of isosurfaces with global illumination. *IEEE Transations on Visualization and Computer Graphics 12* (2006), 186–196. 17

[ZC02] ZHANG C., CRAWFIS R.: Volumetric shadows using splatting. In *IEEE Visualization* (2002), pp. 85–92. 4, 13, 22

[ZC03] ZHANG C., CRAWFIS R.: Shadows and soft shadows with participating media using splatting. *IEEE Transations on Visualization and Computer Graphics 9*, 2 (2003), 139–149. 4, 13, 22

[ZIK98] ZHUKOV S., IONES A., KRONIN G.: An ambient light illumination model. In *EGRW '98: Proceedings of the Eurographics Workshop on Rendering* (1998), pp. 45–55. 7

[ZXC05] ZHANG C., XUE D., CRAWFIS R.: Light propagation for mixed polygonal and volumetric data. In *CGI '05: Proceedings of the Computer Graphics International 2005* (2005), pp. 249–256. 13, 22

Table 1: *Comparison between the different illumination methods.*

Method	Light Source			Scattering	Refresh Time		Memory Consumption	SNR Independent
	Type(s)	Location Constraint	#		Render	Update		
Gradient-based [Lev88]	D, P	None	N	✖	●○○	○○○	○○○	✖
Local Ambient Occlusion [HLY09]	D, P, AMB	-	-	✖	●○○	●●●	●●○	✔
Dynamic Ambient Occlusion [RMSD*08]	D, P, AMB	-	-	S	●○○	●○○	●●●	✔
Half Angle Slicing [KPH*03]	D, P	Outside	1	S,M	●●○	○○○	●○○	✔
Directional Occlusion [SPH*09]	P	Head	1	S	●●○	○○○	●○○	✔
Multidirectional Occlusion [vPBV10]	P	View Hemisphere	N	S,M	●●●	○○○	●○○	✔
Deep Shadow Mapping [HKSB06]	D, P	Outside	1	S,M	●○○	●●○	●●○	✖
Image Plane Sweep [SYR11]	D, P	None	1	S,M	●●○	○○○	●○○	✔
Shadow Splatting [ZC02, ZC03, ZXC05]	D, P, A	Outside	1	S,M	●○○		●○○	✔
Piecewise Integration [HLY08]	D, P	None	1	S	●○○	●●○	●●●	✔
Shadow Volume Propagation [BR98, RDRS10b, RDRS10a]	D, P, A	None	1	S,M	●○○	●●○	●●●	✔
Summed Area Table 3D [DVND10]	AMB	-	-	✖	●○○	●●●	●●○	✔
Extinction-based Shading [SMP11]	D,P,A, AMB	None	N	S	●○○	●●○	●●○	✔
Spherical Harmonics [Rit07, LR10, KJL*12]	D, P, A, T, AMB	Outside	N	S,M	●○○	●●●	●●●	✔
Monte Carlo ray-tracing [KPB11]	D, P, A, T, AMB	None	N	S,M	●●●	●●●	●○○	✔

Legend	
	Light Types: D (Directional), P (Point), A (Area), T (Textured), AMB (Ambient). Light Source #: Number of supported light sources, - (not applicable) N (infinite). Scattering: ✖ (No), S (Single), M (Multiple). ○○○: No quantity of rendering time or memory usage. ●●●: High quantity of rendering time or memory usage. Update Trigger: (Light Source Update), (Transfer Function Update). SNR Independent: ✖ (Sensitive to noisy data) ✔ (Insensitive to noisy data).