DOI: 10.1111/cgf.12252

COMPUTER GRAPHICS *forum*
Volume 33 (2014), number 1 pp. 27–51

A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering

Daniel Jönsson, Erik Sundén, Anders Ynnerman and Timo Ropinski

Scientific Visualization Group, Linköping University
{daniel.jonsson, erik.sunden, anders.ynnerman, timo.ropinski}@liu.se

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 shading and scattering effects. In this survey, we review and classify the existing techniques for advanced volumetric illumination. The classification will be conducted based on their technical realization, their performance behaviour 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.

Keywords: volume rendering, rendering, volume visualization, visualization, illumination rendering, rendering

ACM CCS: I.3.3 [Computer Graphics]: Picture/Image GenerationVolume Rendering

1. Introduction

Interactive volume rendering as a sub-domain 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 real-world use 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 absorbed locally at each sample point processed during rendering. This result 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 simulation of 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]. Due to the wide range of techniques and tradeoffs it can be hard for both application developers and new researchers to know which technique to use in a certain scenario, what has been done and what the challenges are within the field. Therefore, we provide a survey of techniques dealing with volumetric illumination, which allows 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.

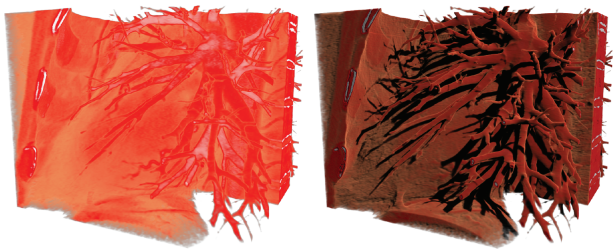


Figure 1: *Comparison of a volume rendered image using the local emission absorption model [Max95] (left), and the global half angle slicing technique [KPHE02] (right). Apart from the illumination model, all other image parameters are the same.*

- 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) 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, seen in Table 1, for advanced volumetric illumination with respect to their applicability, which we derive from the illumination capabilities, the performance behaviour 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 pre-computations 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 adopting a classification which takes into account the technical realization, performance behaviour as well as the supported illumination effects. Properties considered in the classification are for instance, the usage of preprocessing, because pre-computation-based techniques are usually not applicable to large volumetric data sets as the pre-computed 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 paper 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 paper may skip Section 2 and directly proceed to Section 3, where we classify the covered techniques. We then propose usage guidelines in Section 4, which motivates the selected criteria and can help the reader in comparing the different techniques. These guidelines have been formulated with the goal to support an application expert choosing the right algorithm for a specific use case scenario. Based on the classification, Section 5 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 6. Furthermore, based on the observations made in Section 6 and the limitations identified throughout the paper, we will present future challenges in Section 7. Finally, the paper 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

$L(\vec{x}, \vec{\omega}_o)$	Radiance scattered from $\vec{x}$ into direction $\vec{\omega}_o$.
$L_i(\vec{x}, \vec{\omega}_i)$	Radiance reaching $\vec{x}$ from direction $\vec{\omega}_i$.
$L_e(\vec{x}, \vec{\omega}_o)$	Radiance emitted from $\vec{x}$ into direction $\vec{\omega}_o$.
$s(\vec{x}, \vec{\omega}_i, \vec{\omega}_o)$	Shading function used at position $\vec{x}$. Models radiance coming from direction $\vec{\omega}_i$ and scattered into direction $\vec{\omega}_o$.
$\sigma_a(\vec{x})$	Absorption coefficient at $\vec{x}$.
$\sigma_s(\vec{x})$	Scattering coefficient at $\vec{x}$.
$\sigma_t(\vec{x})$	Extinction coefficient at $\vec{x}$. Sum of absorption and out-scattering.
$T(\vec{x}_i, \vec{x}_j)$	Transparency between position $\vec{x}_i$ and position $\vec{x}_j$.
L_l	Initial radiance as emitted by the light source.
$L_0(\vec{x_0}, \vec{\omega}_o)$	Background intensity.

The simplest form of the volume rendering integral describes the attenuation of radiance from all samples $\vec{x}'$ along the ray $\vec{\omega}_o$ and only incorporates emission and absorption, such that it can be written as:

$$L(\vec{x}, \vec{\omega}_o) = L_0(\vec{x_0}, \vec{\omega}_o) \times T(\vec{x_0}, \vec{x}) + \int_{\vec{x_0}}^{\vec{x}} \sigma_a(\vec{x}') \times L_e(\vec{x}', \vec{\omega}_o) \times T(\vec{x}', \vec{x}) d\vec{x}'. \quad (1)$$

While $L_0(\vec{x_0}, \vec{\omega}_o)$ depicts the background energy entering the volume, $L_e(\vec{x}', \vec{\omega}_o)$ and $\sigma_a(\vec{x}')$ describes the radiance at position $\vec{x}'$ that is emitted and absorbed into direction $\vec{\omega}_o$ inside the volume, and $T(\vec{x}_i, \vec{x}_j)$ is dependent on the optical depth and describes how

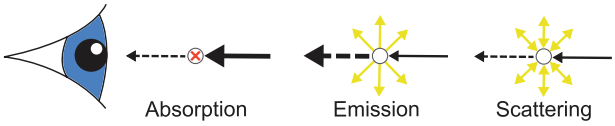


Figure 2: *Before reaching the eye, radiance along a ray may change due to absorption, emission or scattering.*

radiance is attenuated when travelling through the volume:

$$T(\vec{x}_i, \vec{x}_j) = \mathrm{e}^{-\int_{\vec{x}_i}^{\vec{x}_j} \sigma_t(\vec{x}')d\vec{x}'}, \tag{2}$$

where $\sigma_t(\vec{x}') = \sigma_a(\vec{x}') + \sigma_s(\vec{x}')$ is the extinction coefficient, which describes the amount of radiance absorbed and scattered out at position $\vec{x}'$ (see Figure 2). According to Max, the standard volume rendering integral simulating absorption and emission can be extended with these definitions to support single scattering (shadows):

$$L(\vec{x}, \vec{\omega}_o) = L_0(\vec{x}_0, \vec{\omega}_o) \times T(\vec{x}_0, \vec{x}) + \int_{\vec{x}_0}^{\vec{x}} (\sigma_a(\vec{x}') \times L_e(\vec{x}', \vec{\omega}_o) + \sigma_s(\vec{x}') \times L_{\mathrm{ss}}(\vec{x}', \vec{\omega}_o)) \times T(\vec{x}', \vec{x})d\vec{x}'. \tag{3}$$

Here, $L_{\mathrm{ss}}(\vec{x}', \vec{\omega}_o)$ represents the radiance scattered into direction $\vec{\omega}_o$ from all incoming directions $\vec{\omega}_i$ on the unit sphere towards position $\vec{x}'$:

$$L_{\mathrm{ss}}(\vec{x}', \vec{\omega}_o) = \int_\Omega s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o) \times L_i(\vec{x}', \vec{\omega}_i)d\vec{\omega}_i, \tag{4}$$

where $s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o)$ is the material shading function, which describes how much radiance coming from direction $\vec{\omega}_i$ that is scattered into direction $\vec{\omega}_o$ (see Section 2.1). When simplifying the above equation to only deal with one light source at position $\vec{x}_l$, the integral over all directions vanishes and $L_i(\vec{x}', \vec{\omega}_i)$ is the attenuated incident radiance L_l travelling through the volume, defined as

$$L_i(\vec{x}', \vec{\omega}_i) = L_l \times T(\vec{x}_l, \vec{x}'), \tag{5}$$

where $\vec{\omega}_i = \vec{x}_l - \vec{x}'$.

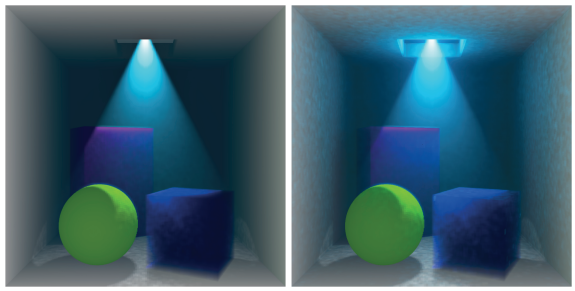
The single scattering formulation in Equation (3) can be extended to support multiple scattering by replacing $L_{\mathrm{ss}}(\vec{x}', \vec{\omega}_o)$ with a term that recursively evaluates the incoming light (see Figure 3 for a comparison):

$$L_{\mathrm{ms}}(\vec{x}', \vec{\omega}_o) = \int_\Omega s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o) \times L(\vec{x}', \vec{\omega}_i)d\vec{\omega}_i. \tag{6}$$

Since volumetric data is discrete, the volume rendering integral is in practice approximated numerically, usually by exploiting a Riemann sum.

2.1. Material scattering

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.



(a) Single scattering (b) Multiple scattering

Figure 3: *Cornell box with a blue transparent medium rendered with and without multiple scattering. Multiple scattering causes photons to bounce to the walls and inside the the blue medium. (Images taken from [JKRY12].)*

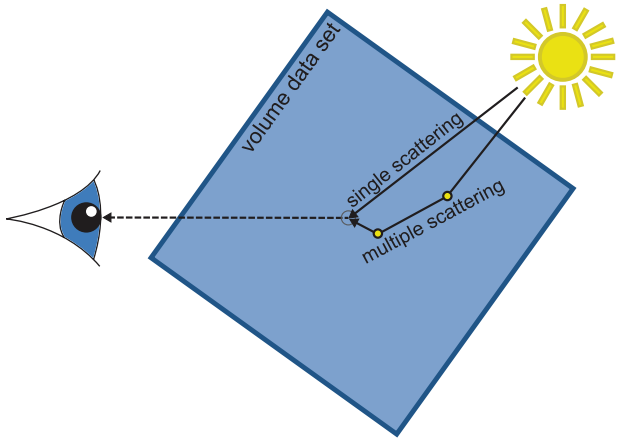


Figure 4: *Single scattering accounts for light that scatters at one location towards the eye, while multiple scattering involves more than one bounce before travelling towards the eye.*

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 colour 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 (see Figure 4). To generate realistic images, both *single* (direct light) and *multiple* (indirect light) scattering events have to be taken into account. While in classical polygonal computer graphics, light reflection on a surface is often modelled using bidirectional reflectance distribution functions (BRDFs), in volume rendering phase functions are used to model the scattering behaviour. 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

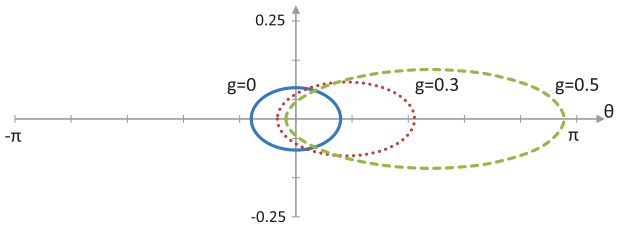


Figure 5: *The Henyey–Greenstein phase function plotted for different anisotropy parameters g.*

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 5 shows the Henyey–Greenstein phase function model for different anisotropy parameters g. In order to incorporate both BRDFs and phase functions into the direct volume rendering equation, we use a loose definition for the shading $s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o)$, which can be a phase function, BRDF, or a combination of both.

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 5.

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 to area and textured light sources. Since several algorithms focus on ambient visibility computation, we consider also an ambient light source which is omnidirectional 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. The discussed algorithms also vary with respect to the light interaction and propagation inside the volume. While all algorithms support single scattering through either local or global light propagation, though at different frequencies producing soft to hard shadows, multiple scattering is not supported by all approaches. We therefore differentiate between local and global scattering as well as single and multiple scattering. 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 tradeoff between rendering performance and memory consumption. The two most extreme cases on this scale are entire pre-computation of the illumination information, and recomputation on the fly for each frame. We will compare the techniques covered within this paper with respect to this tradeoff 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 pre-compute 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 geometry. 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 neighbourhood around a voxel, second *slice-based* techniques which 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

Figure 6: *A visual comparison of 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.*

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 6. 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.

4. 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 of each algorithm. 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 25. Note that distances between methods in Figure 25 are not quantifiable and is only there to get a rough estimate of how they behave relative to each other.

Figure 25(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 multiple scattering as obtained from multiple area light sources. The y-axis in Figure 25(a) on the other hand depicts which algorithmic rendering paradigm that is supported. Thus, Figure 25(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 25(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 can achieve good performance in both aspects. When selecting techniques in Figure 25(a) based on the illumination complexity and the underlying rendering paradigm, Figure 25(b) can be used to compare them regarding their scalability.

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 paper. 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. For memory consumption the zero to three filled circles represents no additional memory, a few additional 2D images, an additional volume smaller than the intensity volume, and memory consumption equal or greater than the intensity volume is required, respectively. An additional icon represents if illumination information needs to be updated when changing the lighting or

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	S (local)	●○○	○○○	○○○	✖
Local Ambient Occlusion [HLY09]	D, P, AMB	None	N	S (local)	●○○	●●●	●●○	✔
Dynamic Ambient Occlusion [RMSD*08]	D, P, AMB	None	N	S	●○○	●○○	●●●	✔
Half Angle Slicing [KPH*03]	D, P	Outside	1	S,M	●●○	○○○	●○○	✔
Directional Occlusion [SPH*09]	P	Head	1	S,M	●●○	○○○	●○○	✔
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	●○○	○○○	●○○	✔
Historygram Photon Mapping [JKRY12]	D, P, A, T	None	N	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	-	-	S	●○○	●●●	●●○	✔
Extinction-based Shading [SMP11]	D,P,A, AMB	None	N	S	●○○	●●○	●●○	✔
Light Propagation Volumes [ZM13]	D,P	None	N	S, M	●○○	●●○	●●●	✔
Spherical Harmonics [Rit07, LR10, KJL*12]	D, P, A, T, AMB	Outside	N	S,M	●○○	●●●	●●●	✔
Monte Carlo ray-tracing [KPB12]	D, P, A, T, AMB	None	N	S	●●●	●●●	●○○	✔
Legend	Light Types: D (Directional), P (Point), A (Area), T (Textured), AMB (Ambient). Light Source #: Number of supported light sources, - (not applicable), N (Many). Scattering: 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).							

the transfer function. Thus, the algorithm performance for frame rendering and illumination updates becomes clear.

5. Algorithm Description

In this section, we will describe all techniques covered within this paper 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.

5.1. Local region-based techniques

As the name implies, the techniques described in this sub-section perform the illumination computation based on a local 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 instance using finite difference gradient operators, only adjacent voxels into account. Other techniques, such as 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 performed 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 7 shows a comparison of gradient-based volumetric shading and the application of the standard emission and absorption model. As 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 $|\bigtriangledown \sigma_t(f(\vec{x}))|$ denoted $\vec{N}$, where $f(\vec{x})$ is the intensity value given at

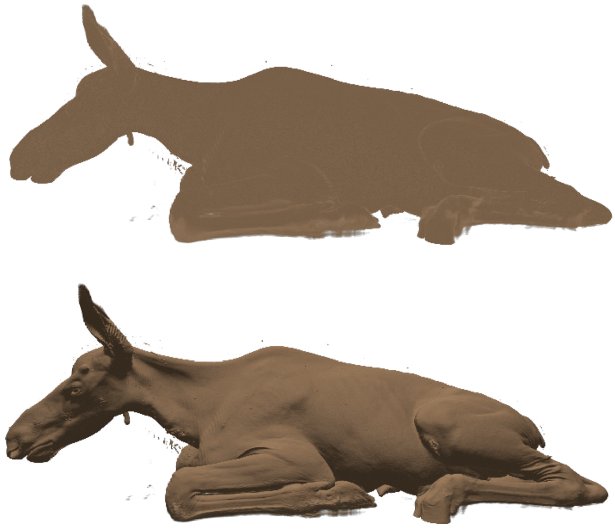


Figure 7: *A CT scan of an elk rendered with emission and absorption only (top), and with gradient-based diffuse shading [Lev88] (bottom). Due to the incorporation of the gradient, surface structures emerge (bottom).*

position $\vec{x}$. Thus, we can define the shading function as:

$$s_{\text{phong}}(\vec{x},\vec{\omega}_i,\vec{\omega}_o) = s_{\text{amb}}(\vec{x}) + s_{\text{diff}}(\vec{x}) \times \vec{N} \times \vec{\omega}_i + s_{\text{spec}}(\vec{x}) \times \left(\vec{N} \times \frac{\vec{\omega}_i + \vec{\omega}_o}{2}\right)^{\alpha}. \quad (7)$$

The material ambient, diffuse and specular components at location $\vec{x}$ are represented by $s_{\text{amb}}(\vec{x})$, $s_{\text{diff}}(\vec{x})$, and $s_{\text{spec}}(\vec{x})$, respectively. They are often referred to as k_{a}, k_{d} and k_{s} in the computer graphics literature. The ambient term approximate indirect illumination effects, which ensure that voxels with gradients pointing away from the light source do not appear pitch black. In this case s_{amb} is simply a constant but more advanced techniques, which include ambient occlusion, are covered below. 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 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 $\bigtriangledown \sigma_t(f(\vec{x}))$ is used in the computation of the diffuse and specular parts, 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 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.

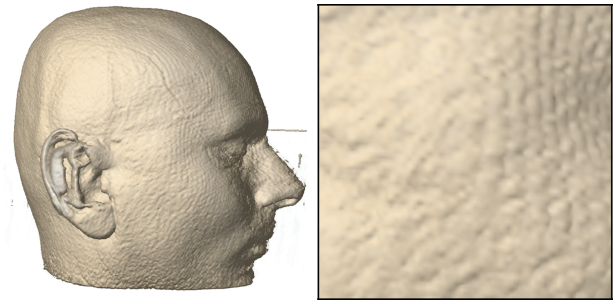


Figure 8: *Gradient-based shading applied to a MRI data set (left). As it can be seen in the close-up, noise is emphasized in the rendering (right).*

Since these techniques are beyond the scope of this paper, 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 or 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 pre-computed and stored in a gradient volume accessed during rendering. No additional memory is required when this pre-computation is not performed. 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 recommended for modalities suffering from a low SNR, such as magnetic resonance imaging (MRI) (see Figure 8), 3D ultrasound or seismic data [PBVG10].

Local ambient occlusion [HLY07] is a local approximation of ambient occlusion, which considers the voxels in a neighbourhood 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 $s_{amb}(\vec{x})$ in Equation (7) by:

$$s_{\text{amb}}(\vec{x}) = \int_{\Omega} \int_{\delta(\vec{x},\vec{\omega}_i,a)}^{\delta(\vec{x},\vec{\omega}_i,R_\Omega)} g_A(\vec{x}') \times T(\vec{x}', \delta(\vec{x},\vec{\omega}_i,a)) d\vec{x}' d\vec{\omega}_i, \tag{8}$$

where $\delta(\vec{x},\vec{\omega}_i,a)$ offsets the position by a along the direction $\vec{\omega}_i$, to avoid self occlusion. R_Ω specifies the radius of the local neighbourhood 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 neighbourhood Ω around a position $\vec{x}'$ (see Figure 9).

Different light contribution functions $g_A(\vec{x}')$ 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.

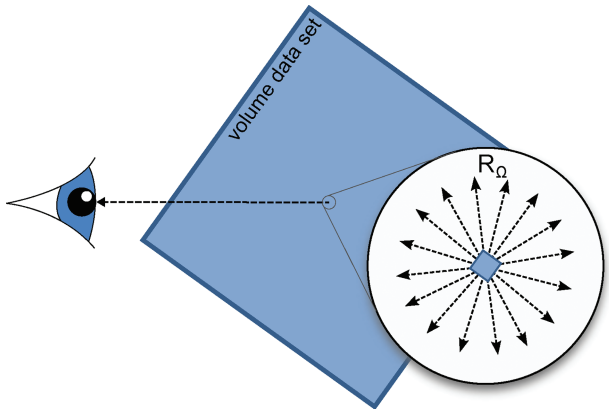


Figure 9: *Local ambient occlusion is computed by, for each voxel, integrating the occlusion over all directions Ω within a radius R.*

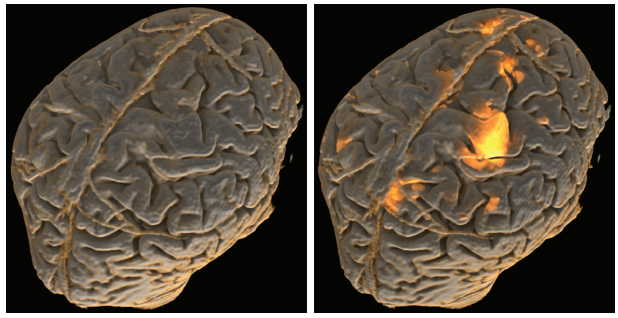


Figure 10: *An application of local ambient occlusion in a medical context. Usage of the occlusion term allows the 3D structure to be conveyed (left), while multi-modal emissive lighting allows fMRI to be integrated as an additional modality (right). (Images taken from [NEO*10].)*

Smoother results can be achieved, when the ambient light source is considered volumetric. This result in adding a fractional light emission at each sample point:

$$g_A(\vec{x}') = \frac{1}{R_\Omega - a} + L_e(\vec{x}', \vec{\omega}_o), \tag{9}$$

where $L'_e(\vec{x},\vec{\omega}_o)$ is a spatially varying emission term that can be used for different tissue types. An application of the occlusion only illumination is shown in Figure 10 (left), while Figure 10 (right) shows the application of the additional emissive term, which is set proportional to the signal strength of a co-registered 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 multi-resolution approach [LLY06]. This flat blocking data structure represents different parts of the volume at different levels of detail. Thus, empty homogeneous regions can be stored at a very low level of detail, which requires less sampling operations when computing $s_{\text{amb}}(\vec{x})$.

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 multi-resolution data structure needs to be updated whenever the transfer function has been changed. The memory size of the multi-resolution 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, colour 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 neighbourhood of $\vec{x}$, dynamic ambient occlusion uses a pre-computation stage, where intensity distributions are computed for each voxel's neighbourhood. The main idea of this approach is to modulate these intensity distributions with the transfer function during rendering, such that integration over the modulated result allows the occlusion to be obtained. 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 containing a cluster ID for each voxel is generated, which associates the voxel with the corresponding row storing the local histogram in the 2D texture (see Figure 11).

During rendering, the pre-computed information is used to support four different illumination effects: ambient occlusion, colour 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 colour to produce the final colour. To integrate ambient occlusion into an isosurface-based volume renderer, the gradient-based shading formulation described above is modulated, such that the ambient intensity $s_{\text{amb}}(\vec{x})$ is derived from the pre-computed data:

$$s_{\text{amb}}(\vec{x}) = 1.0 - O_{\text{env}}(\vec{x}), \tag{10}$$

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

$$O_{\text{env}}(\vec{x}) = \frac{1}{\frac{2}{3}\pi {R_\Omega}^3} \sum_{0 \le j < 2^b} \sigma_\alpha(j) \times LH_j(\vec{x}). \tag{11}$$

Here, R_Ω depicts the radius of the region, for which the local histograms LH have been pre-computed. b is the data bit depth

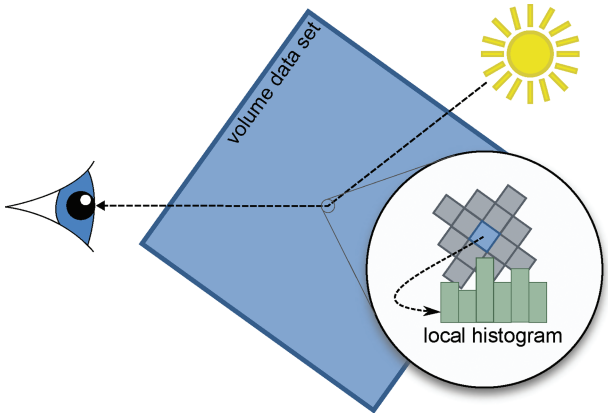


Figure 11: *Dynamic ambient occlusion computes and stores an index to a clustered histogram that is based on the local neighbourhood for each voxel.*

and thus $2b$ is the number of bins of the local histogram. The bins LH_j are modulated based on the opacity $\sigma_\alpha(j)$ assigned to each intensity j. This operation is performed through texture blending on the GPU, and thus allows the occlusion information to be updated in real-time. To apply this approach to direct volume rendering, the environmental colour E_{env} is derived in a similar way as the occlusion factor O_{env}:

$$E_{\text{env}}(\vec{x}) = \frac{1}{\frac{2}{3}\pi {R_\Omega}^3} \sum_{0 \le j < 2b} \sigma_\alpha(j) \times \sigma_{\text{rgb}}(j) \times LH_j(\vec{x}), \tag{12}$$

where $\sigma_{\text{rgb}}(j)$ represents the emissive colour of voxels having the intensity j. During rendering, the current voxel's diffuse colour coefficient k_d is obtained by blending between the transfer function colour σ_{rgb} and E_{env} with using the occlusion factor O_{env} as blend factor. Thus, colour bleeding is simulated, as the current voxel's colour 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 pre-computed 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 colour 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 diffuse and specular contributions. Since the pre-computed 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 low, since the histogram modulation is performed through texture blending. For the pre-computation 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 pre-computation times by a factor of up to 10 [MR10]. During pre-computation, one additional scalar cluster lookup volume as well as the 2D histogram texture is 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 neighbourhoods.

5.2. Slice-based techniques

The slice-based techniques covered in this sub-section 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 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] and later extended to support phase functions and spatially varying indirect extinction for the indirect light in [KPH*03]. 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 multiple scattering within interactive applications. The direct lighting is modelled using the equation supporting a single light source, Equation (5). To include multiple scattering, a directed diffusion process is modelled through an angle dependent blur function. Thus, the multiple scattering using one light source at position x_l with direction $\vec{\omega}_l$ is rewritten as:

$$L_{\text{ms}}(\vec{x}', \vec{\omega}_o) = L_l \times p(\vec{x}', \vec{\omega}_l, \vec{\omega}_o) \times T_{\text{ss}}(\vec{x}_l, \vec{x}') + L_l \times T_{\text{ms}}(\vec{x}_l, \vec{x}') Blur(\theta), \quad (13)$$

where $p(\vec{x}', \vec{\omega}_l, \vec{\omega}_o)$ is the phase function, $T_{\text{ss}}(\vec{x}_l, \vec{x}')$ and $T_{\text{ms}}(\vec{x}_l, \vec{x}')$ denotes the extinction term used for single and multiple scattering, respectively, and θ is a user defined cone angle. Although not entirely obvious from the function name, $Blur(\theta)$ is a function that approximates the indirect incoming radiance within the cone angle. In addition to this modified equation, which approximates forward 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 $C(\vec{x}, \vec{\omega}_o)$ at $\vec{x}$ is computed as:

$$C(\vec{x}, \vec{\omega}_o) = \sigma_a(\vec{x}) \times L_e(\vec{x}, \vec{\omega}_o) \times (1 - S(\vec{x})) + f_{\text{BRDF}}(\vec{x}, \vec{\omega}_l, \vec{\omega}_o) \times S(\vec{x}), \quad (14)$$

where $f_{\text{BRDF}}(\vec{x}, \vec{\omega}_o, \vec{\omega}_l)$ is the BRDF used for the surface. $C(\vec{x}, \vec{\omega}_o)$ is then used in the volume rendering integral as follows:

$$L(\vec{x}, \vec{\omega}_o) = L_0(\vec{x}_0, \vec{\omega}_o) \times T_{\text{ss}}(\vec{x}_0, \vec{x}) + \int_{\vec{x}_0}^{\vec{x}} C(\vec{x}', \vec{\omega}_o) \times \sigma_s(\vec{x}') \times L_{\text{ms}}(\vec{x}', \vec{\omega}_o) \times T_{\text{ss}}(\vec{x}', \vec{x})\, \mathrm{d}\vec{x}'. \quad (15)$$

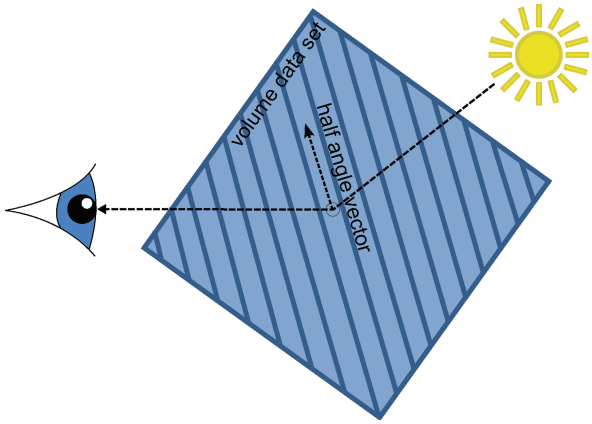


Figure 12: *Half angle slicing performs texture slicing in the orthogonal direction to the half angle between the view direction and the light source.*

The directional nature of the indirect lighting in $L_{\text{ms}}(\vec{x}', \vec{\omega}_o)$ supports an implementation where the rendering and the illumination computation are synchronized. Kniss *et al.* achieve this by exploiting the half angle vector seen in Figure 12. This technique chooses 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 behaviour, half angle slicing is, as all techniques described in this sub-section, bound to the slicing rendering paradigm. It supports one directional or point light source located outside the volume, and can produce shadows having hard, well-defined borders (see Figure 1 (right)). By integrating a directed diffusion process in the indirect lighting computation, forward directed 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 interactive frame rates to be achieved. 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 in $s(\vec{x}, \vec{\omega}_i, \vec{\omega}_o)$ to propagate occlusion information during the slice traversal from front to back:

$$s(\vec{x}, \vec{\omega}_i, \vec{\omega}_o) = \begin{cases} 0 & \text{if } -\vec{\omega}_i \times \vec{\omega}_o < \cos(\theta) \\ \dfrac{1}{2\pi \times (1 - \cos(\theta))} & \text{otherwise.} \end{cases}, \quad (16)$$

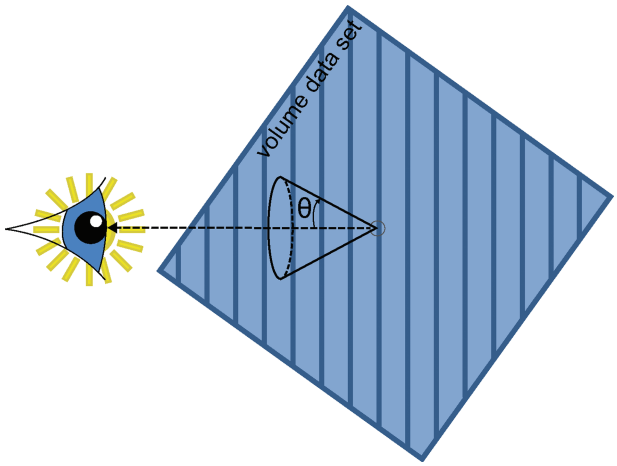


Figure 13: *Directional occlusion shading fixes the light source to the location of the camera and uses a back peaked phase function defined by the angle θ to propagate occlusion information.*

where $\vec{\omega}_o$ is the ray towards the eye and the $-\vec{\omega}_i$'s are lying in the cone centred on $\vec{\omega}_o$ (see Figure 13). 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 $\theta \in [0, \frac{\pi}{2}]$. This backward peaked behaviour is important in order to support synchronized occlusion computation and rendering. Thus, $s(\vec{x}, \vec{\omega}_i, \vec{\omega}_o)$ can be used during the computation of the direct lighting as follows:

$$L_{ss}(\vec{x}', \vec{\omega}_o) = L_l \times \int_\Omega s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o) \times T(\vec{x}_l, \vec{x}') d\vec{\omega}_i, \tag{17}$$

where the background intensity coming from L_l is modulated by $s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o)$, which is evaluated over the contributions coming from all directions $\vec{\omega}_i$ over the sphere. Since $s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o)$ is defined as a backward peaked phase function, in practice the integral only needs to be evaluated in an area defined by the cone angle θ. To incorporate the occlusion-based lighting into the volume rendering integral, Schott *et al.* propose to remove the emission term $L_e(\vec{x}', \vec{\omega}_o) = 0$ from Equation (3):

$$L(\vec{x}, \vec{\omega}_o) = L_0(\vec{x}_0, \vec{\omega}_o) \times T(\vec{x}_0, \vec{x}) + \int_{\vec{x}_0}^{\vec{x}} \sigma_s(\vec{x}') \times L_{ss}(\vec{x}', \vec{\omega}_o) \times T(\vec{x}', \vec{x}) d\vec{x}'. \tag{18}$$

The implementation of occlusion-based shading is similar to standard front-to-back slice rendering. However, 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 soft shadow effects to be generated by approximating multiple scattering from a cone shaped phase function (see Figure 14 (left) and (middle)). The performance is interactive and comparable to half angle slicing but, since no pre-computation is used, illumination updates does not need 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.

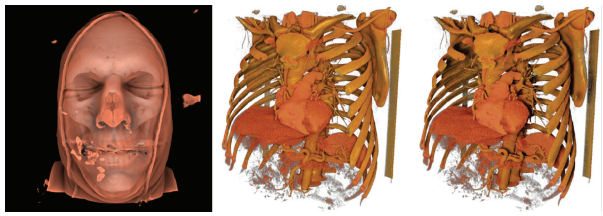


Figure 14: *Directional occlusion shading [SPH*09] integrates soft shadowing effects (left). Multi-directional occlusion shading [vPBV10] additionally supports different lighting positions (middle) and (right). (Images taken from [SPH*09] and [vPBV10], courtesy of Mathias Schott and Veronika Šoltészová.)*

Patel *et al.* [PBVG10] apply concepts similar to directional occlusion shading when visualizing seismic volume data sets. By using the incremental blurring operation modelled through the phase function, they are able to visualize seismic data sets without emphasizing inherent noise in the data. A more recent extension by Schott *et al.* [SMGB12, SMG*13] allows geometry to be integrated and support light interactions between the geometry and the volumetric media.

Multi-directional 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. The directional occlusion shading simplifies computation by restricting the light source direction to be aligned with the viewing direction, and therefore does not allow the light source position to be changed. Multi-directional 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. In their paper, they derive the convolution kernel from the light source position and the aperture θ defining the cone. Due to the directional component introduced by the front-to-back slicing, the tilt angle of the cone is limited to lie in $[0, \frac{\pi}{2} - \theta]$ towards the viewer, which restricts the shape of the filter kernel from generating into hyperbolas or parabolas. The underlying illumination model is the same as for directional occlusion shading, whereas the indirect lighting contribution is modified by exploiting ellipse shaped filter kernels.

As standard directional occlusion shading, multi-directional occlusion shading uses front-to-back slice rendering, which tightly binds it to this rendering paradigm. Since the introduced filter kernels allow light sources to be placed at different positions, multiple light sources are supported. However, due to the directional nature, light sources must be positioned in the viewer's hemisphere. Light scattering capabilities are the same as with directional occlusion shading, i.e. soft shadows through multiple 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 14,

where Figure 14 (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 14 (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.

5.3. 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 to 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.

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 within the range given by the error threshold anymore, a new segment to be approximated by a linear function is started. A small error threshold results in a better approximation, but more layers are needed to represent the shadow function.

A different approach was presented by Jansen and Bavoil [JB10]. They approximate the absorption function using a Fourier series, which works especially well for smooth media such as fog and smoke. The opacity is projected into the Fourier domain and a fixed number of coefficients are used to store the approximation in a Fourier opacity map (FOM). This enables them to get a more accurate approximation at comparable memory consumption and speed compared to the original deep shadow mapping [LV00] approach. However, as Jansen and Bavoil point out in their paper, high density medias may cause ringing artifacts. Therefore, Delalandre *et al.* [DGMF11] improved the FOM method and used it to represent transmittance instead of opacity, which enabled them to quickly evaluate single scattering with projective light sources. Furthermore, they proposed a projective sampling scheme, which take samples uniformly in light space, and also allow geometries to be integrated in the rendering. Gautron *et al.* [GDM11] further expanded on the work of Delalandre *et al.* [DGMF11] by encoding the variation of the extinction along a ray and enabling the use of particle-based media.

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 when using gradient-based shading. Nevertheless, it can also be used to directly modulate the emissive contribution $L_e(\vec{x}, \vec{\omega}_o)$ at $\vec{x}$.

A comparison of deep shadow mapping, which enable semi-transparent occluders, with shadow rays and standard shadow mapping is shown in Figure 15. 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 result 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 and update time 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 or coefficients.

Desgranges *et al.* [DEP05] also propose a gradient-free technique for shading and shadowing that is based on projective texturing.

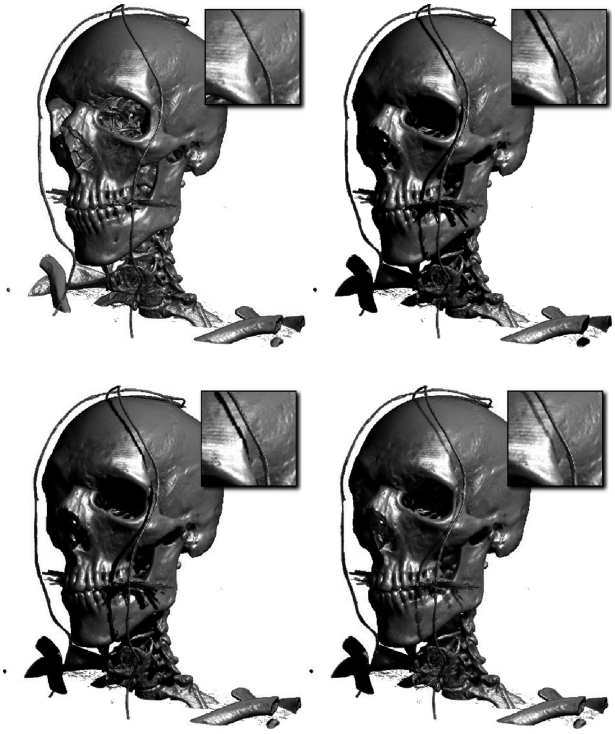


Figure 15: *The visible human head data set rendered without shadows (top left), with shadow rays (top right), with shadow mapping (bottom left) and with deep shadow maps (bottom right). (Images taken from [RKH08].)*

Their approach produces visually pleasing shadowing effects and can be applied interactively.

Image plane sweep volume illumination [SYR11] allows advanced illumination effects to be integrated 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 single scattering with a single light direction, or multiple scattering resulting from a rather high albedo [MC10]. Multiple scattering is simulated by assuming the presence of a high albedo, making multiple scattering predominant and possible to approximate using a diffusion approach:

$$I_{\text{ms}}(\vec{x}, \omega_o) = \nabla_{\text{diff}} \int_{\vec{x}_b}^{\vec{x}} \frac{1}{|\vec{x}' - \vec{x}|} \times L_e(\vec{x}', \vec{\omega}_o) d\vec{x}'. \quad (19)$$

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 lying outside the volume. In practice, the computation of the scattering contributions can be simplified, 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:

$$\begin{aligned} I'_{\text{ss}}(\vec{x}, \omega_o) = {} & L_0(\vec{x}_l, \omega_l) \times T(\vec{x}_l, \vec{x}) \\ & + \int_{\vec{x}''}^{\vec{x}} \sigma_t(\vec{x}') \times L_e(\vec{x}', \vec{\omega}_o) \times T(\vec{x}', \vec{x}) d\vec{x}' \\ & + \int_{\vec{x}_l}^{\vec{x}''} \sigma_t(\vec{x}') \times L_e(\vec{x}', \vec{\omega}_o) \times T(\vec{x}', \vec{x}) d\vec{x}', \quad (20) \end{aligned}$$

and a similar representation for the multiple scattering contribution:

$$\begin{aligned} I'_{\text{ms}}(\vec{x}, \omega_o) = {} & \nabla_{\text{diff}} \int_{\vec{x}''}^{\vec{x}} \frac{1}{|\vec{x}' - \vec{x}|} \times L_e(\vec{x}', \vec{\omega}_o) d\vec{x}' \\ & + \nabla_{\text{diff}} \int_{\vec{x}_b}^{\vec{x}''} \frac{1}{|\vec{x}' - \vec{x}|} \times L_e(\vec{x}', \vec{\omega}_o) d\vec{x}'. \quad (21) \end{aligned}$$

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 can thus only be used within this rendering paradigm. It supports one point or directional light source, and simulates single as well as forward multiple scattering effects. Since all illumination computations are performed directly within a single rendering pass, it does not require any preprocessing and 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 was first proposed 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. Single scattering is supported and this method is therefore based on Equation (3), but uses a filter kernel to splat light information into a shadow volume. The shadow volume is updated in a first splatting pass, during which light information is attenuated as it travels through the volume using Equation (5) for each light source. In the second splatting pass, which is used for rendering, the shadow volume is used to acquire the incident illumination.

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

Zhang and Crawfis [ZC02], [ZC03] present an algorithm which does not require storage of a shadow volume in memory. Their algorithm instead exploits two additional image buffers for handling illumination information (see Figure 16). These buffers are used to attenuate the light during rendering and to add its contribution to the final image. Whenever a light contribution is added to the final

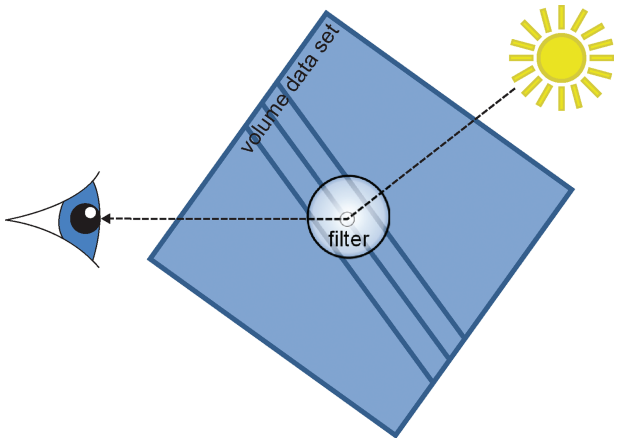


Figure 16: *Shadow splatting uses a filter kernel to splat the light contribution into buffers in a pass sweeping from the light source direction.*

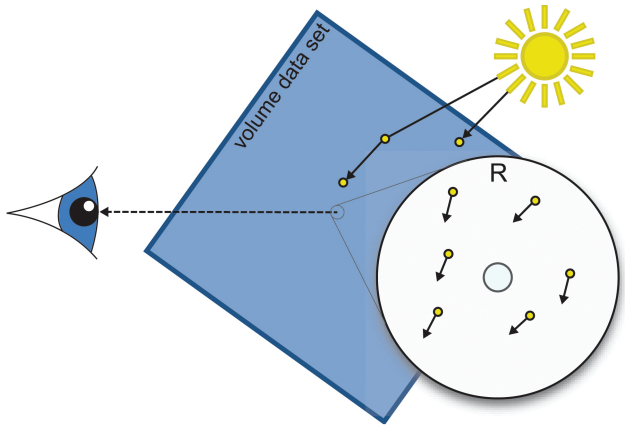


Figure 17: *Photon mapping sends light particles into the scene and then collects all photons within a radius R to approximate the in-scattered radiance.*

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 with multiple point light and distant light sources, as well as textured area lights. While the initial algorithm [ZC02] only supports hard shadows, a later extension integrates soft shadows [ZC03]. The rendering time is approximately doubled with respect to regular splatting when computing shadows. 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. A later extension makes the integration of geometry possible [ZXC05].

Historygram photon mapping [JKRY12] enabled interactive editing of transfer functions by only recomputing the parts of the light transport solution that actually changes. Photon mapping is used as the underlying light transport method, which is based on Equation (3), but with the single scattering term $L_{ss}(\vec{x}', \vec{\omega}_o)$ exchanged with multiple scattering $L_{ms}(\vec{x}', \vec{\omega}_o)$ from Equation (6). The recursive nature of multiple scattering is avoided by approximating the in-scattered radiance with the n energy particles $\Phi_i(\vec{x}', \vec{\omega}_p)$, called photons, within the spherical neighbourhood of radius R:

$$L_{ms}(\vec{x}', \vec{\omega}_o) \approx \frac{1}{\sigma_s(\vec{x}')} \sum_{i=1}^{n} s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o) \frac{\Delta\Phi_i(\vec{x}', \vec{\omega}_i)}{\frac{4}{3}\pi R^3}. \quad (22)$$

The method boils down to a two pass algorithm that first sends photons from the light sources, stores them and then sends view rays from the camera that gathers the transmitted photons according to the equation above (see Figure 17). A hash data structure is used to store and retrieve the photons lying within the radius R of the location $\vec{x}'$. Since both the process of sending and gathering photons is expensive, a concept for reducing these computations, called historygram, is proposed. A historygram, H, is the representable set of parameters that have been used during the computation, and a mapping function $\delta(x)$ is used in order to map all possible parameters into this representable set. A historygram is created by applying the union of the previous historygram, H_i, with the n new mapped parameters x_j, $j = 1..n$:

$$H_{i+1} = H_i \bigcup_{j=1}^{n} \delta(x_j). \quad (23)$$

A binary data structure is used for the historygrams, which enables low memory overhead and hardware instructions to be used when evaluating the mapping function $\delta(x)$ and comparing historygrams. For photon mapping, a historygram represent the data that is used and each photon stores a 64-bit historygram. View rays are divided into segments that also each have a historygram. When the user changes the transfer function, the affected data is encoded into a change historygram and then evaluated against each photon's and view ray segment's associated historygram. A photon or view ray segment is only recomputed if it was affected by the transfer function change. The view ray segments must be recomputed whenever the camera changes, but the photons can be reused as they are not dependent on the camera placement. Jönsson *et al.* [JKRY12] showed that speedups of up to 15 times can be achieved when applying the historygram concept compared to a brute force photon mapper, thus enabling interactive transfer function editing.

This method inherits many properties from the original photon mapping method introduced by Jensen [Jen96]. As such, there is no restriction on the shape, number or placement of light sources. Advanced material effects and mixtures of phase function and BRDF can be used as well as multiple scattering, which makes it possible to create realistic images such as the one in Figure 18. The amount of memory required is highly dependent on the number of photons and view ray segments used. The hash data structure requires memory in addition to the photon data, which consist of position, direction, power and historygram. Each additional view ray segment requires 16MB of GPU memory for a 1024×1024 viewport resolution. Clip plane editing is supported, although not at interactive frame rates.

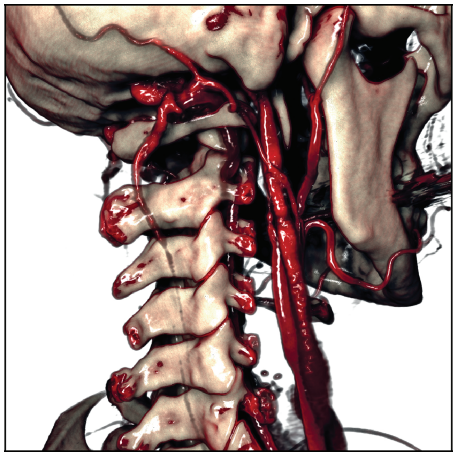


Figure 18: *The neck of a human skull rendered with the historygram photon mapping method, which can handle complex light set ups and multiple scattering. (Images taken from [JKRY12].)*

5.4. Lattice-based techniques

We classify all those approaches that compute the illumination directly based on the underlying grid as lattice-based techniques. 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 survey, we focus on 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 step through the regular grid defining the volumetric data set. The first algorithm realizing this principle was 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 data set 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 centred about the incoming light direction. However, in difference from [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 for multiple scattering in Equation (6) is modified as follows:

$$L_{\text{ms}}(\vec{x}', \vec{\omega}_o) = t(\vec{x}') \times l_i(\vec{x}', \vec{\omega}_l) \times c(\vec{x}', \vec{\omega}_o). \tag{24}$$

where $t(\vec{x}')$ is the transport colour, as assigned through the transfer function, $l_i(\vec{x}', \vec{\omega}_l)$ is the luminance of the attenuated direct light as defined in Equation (5), and $c(\vec{x}', \vec{\omega}_o)$ the chromaticity of the indirect in-scattered light. The transport colour $t(\vec{x}')$ as well as the chromaticity $c(\vec{x}', \vec{\omega}_o)$ are wave length dependent, while the scalar $l_i(\vec{x}', \vec{\omega}_l)$ 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

$$c(\vec{x}', \vec{\omega}_o) = \int_\Omega s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o) \times L_{\text{chrom}}(\vec{x}', \vec{\omega}_i) d\vec{\omega}_i, \tag{25}$$

where Ω is the unit sphere centred around $\vec{x}$ and $L_{chrom}(\vec{x}, \vec{\omega}_i)$ is the chromaticity of Equation (3) with multiple scattering. To allow unidirectional illumination propagation, $s(\vec{x}', \vec{\omega}_i', \vec{\omega}_o)$ is chosen to be a strongly forward peaked phase function:

$$s(\vec{x}', \vec{\omega}_i, \vec{\omega}_o) = \begin{cases} 0 & \text{if } \vec{\omega}_i \times \vec{\omega}_o < \cos(\theta(\vec{x})) \\ (C \times \vec{\omega}_i \times \vec{\omega}_o)^\beta & \text{otherwise.} \end{cases} \tag{26}$$

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. C is a constant, chosen with respect to β, which ensures that the function integrates to one over all angles.

During rendering, the illumination volume is accessed to lookup the luminance value and scattering colour 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 $L_{\text{ms}}(\vec{x}', \vec{\omega}_o)$ 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] focus on direct light, i.e. Equation (3), with a single light source. They propose to discretize Equation (5) by dividing rays into k segments and evaluating the incoming

radiance using:

$$L_i(\vec{x}', \vec{\omega}_i) = L_l \times \prod_{n=0}^{k} T(\vec{x}_n, \vec{x}_{n+1}). \tag{27}$$

Short rays are first cast towards the light source for each voxel and the piecewise segments are stored in an intermediate 3D texture using a multi-resolution data structure [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 in-scattering 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 includes an approximation of first-order in-scattering. Rendering times are low, since only a single extra lookup is required to compute the incoming radiance at the sample position. The radiance must be recomputed as soon as the light source or transfer function changes, which can be performed interactively and progressively. Three additional 3D textures are required for the computations. However, a lower resolution grid with multi-resolution structure is 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:

$$\mathrm{SAT}_{image}(x, y) = \sum_{i=0, j=0}^{x, y} p_{i,j} \tag{28}$$

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 neighbourhood 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 the average depth is larger than the depth of the pixel, then the neighbouring 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 in depths around the corresponding pixel. A halo effect can be achieved by also considering the average depth from the neighbourhood. If the structure in the pixel lies outside the object, a halo colour is rendered which decays with respect to the distance to the object.

Density summed area table [DVND10] utilizes a 3D SAT for ambient occlusion, calculated from the density values stored in the volume. This procedure is view independent, supports incorporation of semi-transparent structures and is more accurate than the 2D image space approach [DYV08]. While only eight 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 a technique by Schlegel *et al*. [SMP11], where a 3D SAT is used to model ambient occlusion, colour bleeding, directional soft shadows and scattering effects by utilizing the summation of the exponential extinction coefficient in the transmittance equation:

$$T(\vec{x}_i, \vec{x}_j) = \mathrm{e}^{-\int_{\vec{x}_i}^{\vec{x}_j} \sigma_t(\vec{x}')\,\mathrm{d}\vec{x}'} \approx \mathrm{e}^{-\sum_{j=i}^{\left|\vec{x}_j - \vec{x}_i\right|/\Delta x} \sigma_{t_j} \Delta x}. \tag{29}$$

Traditionally, this equation has been approximated with a Taylor series expansion, resulting in an α-blending operation. However, current GPUs does not need to perform this approximation, which allow the summation to be split and performed in any order. A 3D SAT can then be used to approximate the sum in the exponent for a set of directions, thereby decreasing the computational cost significantly. Ambient occlusion and colour bleeding is approximated using a number of shells formed by cuboids around the sample point. The extinction sum, $\sigma_{Sh_{i+i}}$, is accumulated for each shell according to

$$\sigma_{Sh_{i+i}} = \sigma_{Sh_i} + (\mathrm{SAT}(Sh_{i+1}) - \mathrm{SAT}(Sh_i)) \times r_{Sh_{i+1}}^{-2}, \tag{30}$$

where Sh_i denotes the shell of index i and $r_{Sh_{i+1}}$ is the radius of the next shell. Schlegel *et al*. [SMP11] reported that three shells are sufficient for a radius that does not exceed 10% of the data set's extent. Directional soft shadows and scattering effects are incorporated by estimating a cone defined by the light source. The approximation of the cone is performed with a series of cuboids (see Figure 19). 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 and the primary axis is defined to be the one with the smallest angle towards the light source. The light sources are not restricted to lie outside of the volume since extinction, and not light propagation, is computed. A selection of renderings using this method is shown in Figure 20.

Extinction-based shading and illumination is not limited to any specific rendering paradigm although it was implemented for ray-casting. The technique can handle point, directional and area light sources. The extinction must be estimated for each additional light source, 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

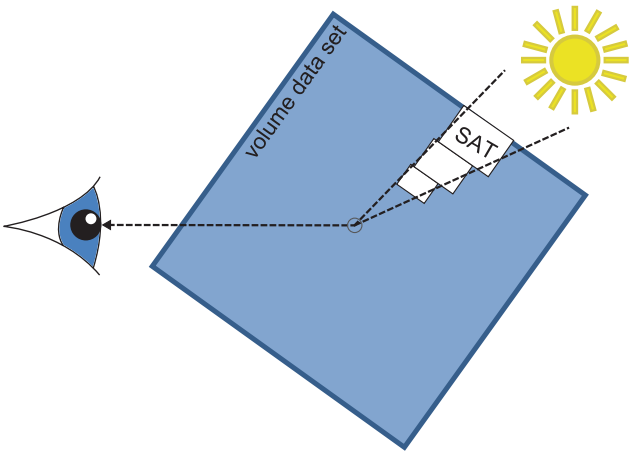


Figure 19: *Extinction-based shading computes the integrated extinction towards the light using summed area tables formed by a series of axis-aligned cuboids (2D for illustration purposes).*

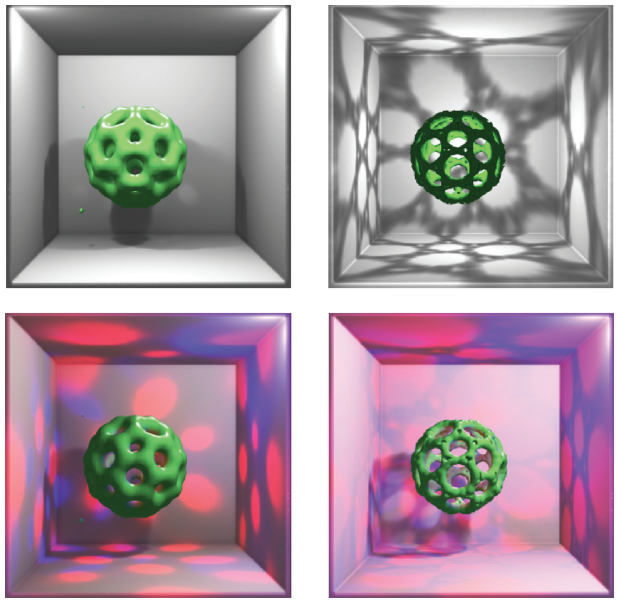


Figure 20: *Extinction-based shading and illumination [SMP11] allows interactive integration of local and global illumination properties and advanced light setups. (Images taken from [SMP11], courtesy of Philipp Schlegel.)*

the volume data while still producing convincing results. It is possible to interactively change clip planes due to the fast computation and lower resolution of the 3D SAT.

Light propagation volumes injects the illumination into a grid and propagates the light according to a convection-diffusion equation. The convection part of the equation describes how direct light propagates inside the volumetric media, while the diffusion part describes how the indirect light is scattered inside the volumetric media.

Geist *et al.* [GRWS04] used a lattice-Boltzmann method to solve the light transport, but their method did not reach interactive frame rates. More recently, Kaplanyan and Dachsbacher [KD10] presented a real-time method using spherical harmonics. They introduced a novel propagation method that only requires light to be propagated among the main axial directions instead of 26 directions in other schemes. However, the method is primarily focused on low-frequency lighting and simulates single-bounce indirect light. Zhang and Ma [ZM13] presented a method that simulates both single and multiple scattering and is able to reach interactive frame rates. Zhang and Ma describe the equation for radiance convection-diffusion, in differential form, as:

$$\frac{\mathrm{d}}{\mathrm{d}t}L(\vec{x}) = -c \times \vec{\omega}_l(\vec{x}) \times \nabla L(\vec{x}) - \sigma_a(\vec{x}) \times L(\vec{x}) + \sigma_s \times \nabla^2 L(\vec{x}), \tag{31}$$

where $L(\vec{x})$ is the radiance at location $\vec{x}$, $\frac{\mathrm{d}}{\mathrm{d}t}L(\vec{x})$ is thus the rate of change of radiance at location $\vec{x}$, c is the speed of light and $\vec{\omega}_l(\vec{x})$ is the normalized light direction at $\vec{x}$. The first two terms describe light leaving and being absorbed at location $\vec{x}$, respectively, which is referred to as convection. The third term, referred to as diffusion, approximates isotropic scattering and can intuitively be understood as light being dissipated within the volumetric media. In order to solve this equation, Zhang and Ma discretize the radiance distribution into a uniform grid and divide the computation into two steps, convection and diffusion, both performed on the GPU. By assuming that the direct light propagation is unique for each grid point the directional part of the radiance can be discarded and only the colour channels need to be stored, thus reducing the storage requirements. The convection step computes direct light propagation for each light source. The radiance grid is first initialized using different boundary conditions for directional and point light sources, essentially injecting light at the boundary of the light volume. In addition to this, point light sources inside the volume are approximated using a Gaussian function and injected into the grid. Once initialized, a first-order upwind scheme is used to propagate the light until a steady state solution is obtained. The result of the propagation of each light source is summed up and then used as an initial estimate in the diffusion step. The diffusion step is solved using the Gauss-Seidel method with time $t = 1$ as the final illumination volume. Once the light has been propagated, the rendering phase uses standard ray-casting with Phong shading and a simple lookup from the illumination volume to get the radiance at the location along the ray, and thus allow real-time frame rates when changing the viewpoint.

Since the method only applies a lookup during rendering it can be combined with other rendering paradigms. It supports multiple dynamically moving directional and point light sources that can be placed anywhere in the scene with interactive updates. However, Zhang and Ma [ZM13] report that only soft shadows are supported due to the use of a first-order upwind scheme. Multiple scattering is supported with the assumption of an isotropic phase function. The light propagation needs to be recomputed as soon as the transfer function or a light source changes, but the computation can be solved in an iterative manner in order to improve the interaction. Low rendering times are achieved by only performing a single lookup is to compute the incoming radiance. Two temporary 3D illumination volumes are needed during the computation in addition to the final illumination volume, each consisting of three channels. However, Zhang and Ma report that plausible results can be achieved using

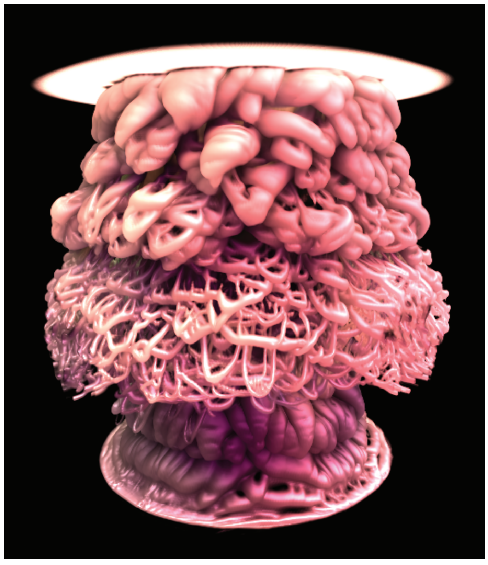


Figure 21: *Rendering of a vortex field from a turbulent flow simulation with an illumination volume of half the size of the original data using light propagation volumes [ZM13], which can handle multiple light sources as well as multiple scattering. (Images taken from [ZM13], courtesy of Yubo Zhang.)*

half the resolution of the original (see Figure 21), which reduces the memory requirements.

5.5. Basis function-based techniques

The group of basis function-based techniques covers those approaches where the light source radiance and transparency (visibility) $T(\vec{x}_i, \vec{x}_j)$, from the position $\vec{x}_i$ to the position $\vec{x}_j$ (usually border of volume), is separated and represented using a basis function. Since radiance and visibility are computed independently, light sources may be dynamically changed without an expensive visibility update, which is the case for many other methods. Furthermore, the light source composition may be complex as they are projected to another basis that is independent of the number of light sources. To this end, the only basis function 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. It is beyond the scope of this paper 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. This being said, we will mention two important properties. The first property 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. 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 was first applied to direct volume rendering by Ritschel [Rit07] 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 multi-resolution 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 multi-resolution 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.

Zhou *et al.* [ZRL*08] presented a method for rendering smoke under low-frequency illumination by decomposing the smoke animation into a spherical harmonics basis representation and a sequence of residual fields. The light computation is then performed using the spherical harmonics representation of the density field. The computed illumination is combined with the high-frequency residual fields during ray marching to get high details in the final rendering.

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 pre-computed 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 coefficients, 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 multi-resolution technique of Ljung *et al.* [LLY06]. The sparse representation of volume data and visibility allows substantial reductions with respect to memory requirements. To also decrease the visibility update time, the two pass approach introduced by Hernell *et al.* [HLY08] is used. First, local visibility is computed over a small neighbourhood by sending rays uniformly distributed on the sphere. Then, piecewise integration of the local visibility is performed using the spherical harmonics representation of Dirac delta functions in the direction of integration in order to compute global visibility. 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 so far. An arbitrary number of dynamic directional light sources are supported, i.e. environment maps. Area

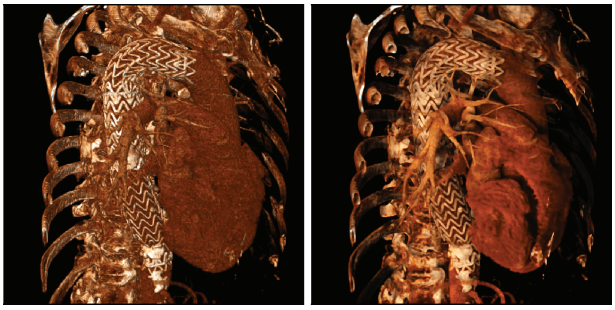


Figure 22: *A CT scan of a human torso rendered with diffuse gradient-based shading (left) and spherical harmonic lighting (right). (Images taken from [KJL*12].)*

Figure 23: *Monte Carlo ray-tracing allows gradient-based material and camera properties to be incorporated, which result in highly realistic images. (Image taken from [KPB12], courtesy of Thomas Kroes.)*

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. A comparison to diffuse gradient-based shading can be seen in Figure 22. 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 inside. 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. The visibility must be recomputed when the transfer function or clip plane changes, which is expensive but can be performed interactively. 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 tradeoff memory size and image quality.

5.6. 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 set-up 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].

Monte Carlo ray-tracing methods have also been applied in the area of interactive volume rendering in order to produce realistic images. 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 single and multiple scattering, whereby interactivity is achieved by constraining the expensive scattering computations to selected surfaces only. However, due to this restriction it is not possible to render translucent materials such as clouds.

More recently, Kroes *et al.* presented an interactive Monte Carlo ray-tracer, which does not limit the scattering to selected boundary surfaces [KPB12]. 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 images showing a high degree of realism to be generated (see Figure 23). To adequately represent both, volumetric and surface-like materials, their approach uses stochastic sampling with the gradient magnitude as threshold to determine if a BRDF or a phase function should be used for material representation. It is restricted to single scattering and uses a progressive approach to allow interactivity. It is shown that the image converges after roughly a half to one second when parameters such as clip plane or transfer function is changed. A minor amount of additional data is required in order to perform filtering and noise removal, but the method is independent of the data set size in terms of memory.

5.7. Isosurface-based techniques

Besides the algorithms discussed above, which can be combined with direct volume rendering and are therefore the main focus of this paper, several approaches exist that allow advanced illumination when visualizing isosurfaces. While dealing with direct volume rendering involves a transfer function that may extract several structures having different extinction and optical properties, when dealing with the simplest form of isosurface rendering, only a single isosurface with homogeneous optical properties is visible. This drastically reduces the combinations of materials that interact with each other and produce complex lighting effects. 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 neighbouring voxels, was the first approach addressing advanced

illumination for isosurfaces extracted from volume data. This method has a pre-computation step where the vicinity of each voxel is analyzed and the resulting value, which represents the occlusion of the voxel, is stored in a shading texture that can be accessed during rendering. Wyman *et al*. [WPSH06] and Beason *et al*. [BGB*06] focus on a spherical harmonics-based pre-computation, 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 Beason have demonstrated how to decouple illumination sampling from isosurface generation, in order to achieve sophisticated isosurface illumination results [BB09].

6. Perceptual Impact

In addition to 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*. 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 showed 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.

Šoltészová *et al*. also evaluated the perception of surface slant for surfaces with Lambertian shading and found that there is a systematic distortion [vTPV12]. They create a prediction model that is used to alter the normal or gradient in the shading of the surface slant such that it is perceived as the ground truth. This improves the perception of the surface slant when the angle is between 40° and 60°. The difference before and after applying their corrective model can be seen in Figure 24.

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 showed that advanced volumetric illumination models make a difference when it is necessary to assess depth or size of volumetric objects 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.

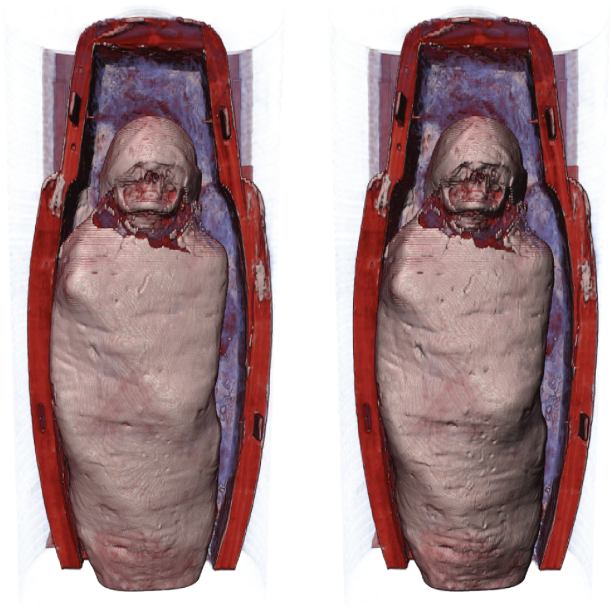


Figure 24: *A CT scan of a mummy displaying a comparison between Lambertian shading with (left), and without (right) perceptual gradient correction. (Image taken from [vTPV12], courtesy of Veronika Šoltészová.)*

In terms of light source placement, Langer *et al*. [LB00] showed that local shape perception increases if a directional light source is angled slightly to the left or right above the viewing direction (11°) compared to placing it below the viewing direction. They further showed that diffuse light produced as good shape perception as the light coming from upper right or left. Caniard *et al*. [CF07] showed that shape perception is heavily dependent on light source position when using a single point light source under local illumination. Creating more complex light set-ups can increase the shape perception [HM03]. As noted before, Wanger *et al*. [Wan92] showed that the sharpness of shadows increase the accuracy of matching shapes. Interreflections (multiple scattering) was shown by Madison *et al*. [MTK*01] to be an important cue, in addition to shadows, for determining the contact of objects. Inter-reflections were also used by Weigle and Banks [WB08], which showed that physically based illumination can in some cases increase the perception of 3D flow-field geometry.

7. Future Challenges

Though the techniques presented in this paper allow realistic volume rendered images to be generated 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*. [KPB12] and Jönsson *et al*. [JKRY12] can be considered as first valuable steps into this direction. One important ingredient

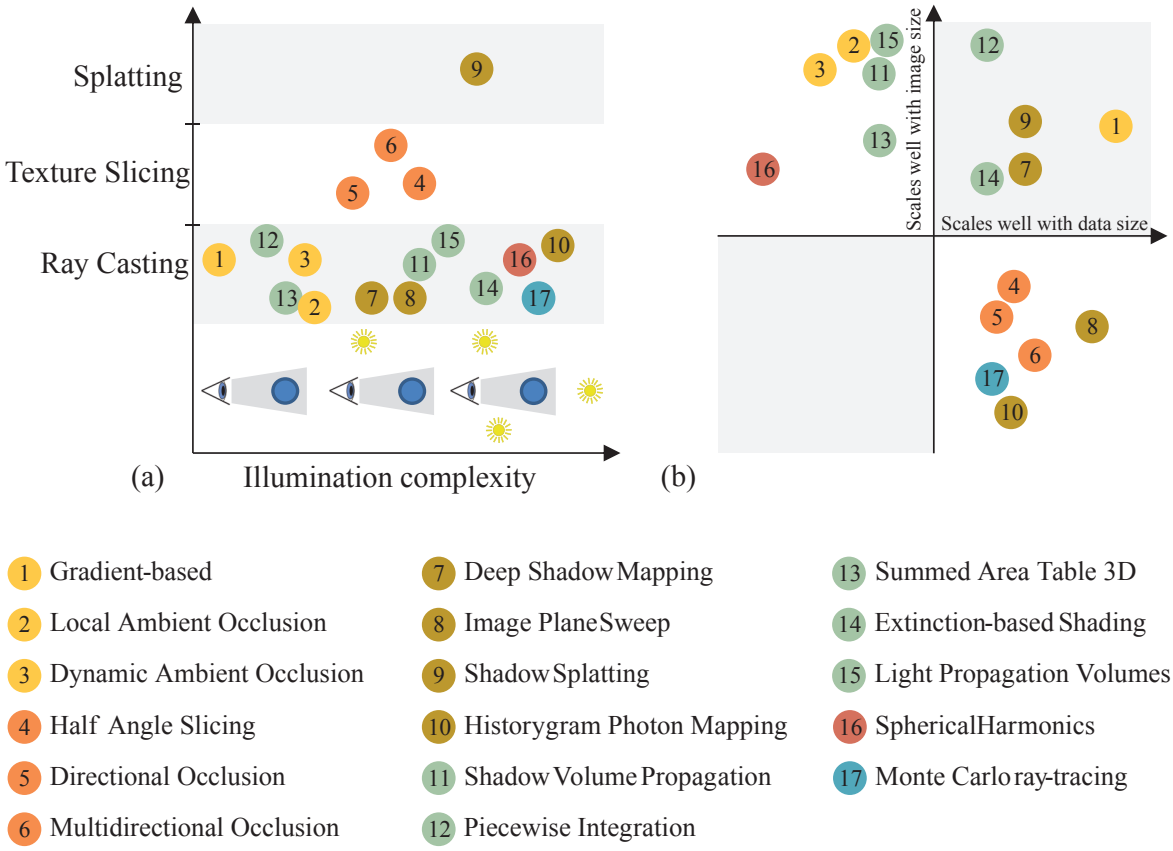


Figure 25: *In (a), the illumination complexity of a method is shown in relation to the underlying rendering paradigm, whereby the illumination complexity increases from left to right. The type of illumination complexity a method can handle is depicted on the horizontal axis. Starting with local shading to the left, advancing to a single static light source and finally ending up with dynamic light sources, multiple scattering and advanced materials. 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. Note that distances between methods are not quantifiable and should more be seen as rough guidelines.*

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 these lines, also the integration of volumetric and polygonal information needs more consideration. While the approaches by Schott *et al.* [SMGB12, SMG*13] provides 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. Here, the illumination computation time increases and it is not possible to store large intermediate data sets due to memory limitations. Although memory capacity on GPUs has increased during the last couple of years, the speed at which data can be transferred for calculations has not increased at the same rate. Multi-resolution data structures [WGLS05, LLY06] and explicit caches [JGY*12] could alleviate these problems but more work in relation to volumetric illumination is needed.

8. Conclusions

Within this paper, 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 using a uniform mathematical notation. Thus, the paper can be used as both a reference for the existing techniques and 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

This work was supported through grants from the Excellence Center at Linköping and Lund in Information Technology (ELLIIT), the

Swedish Research Council through the Linnaeus Center for Control, Autonomy and Decision-making in Complex Systems (CADICS), the Swedish Research Council (VR, grant 2011-4113), and the Swedish e-Science Research Centre (SeRC). 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.

[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.

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

[BR98] Behrens U., Ratering R.: Adding shadows to a texture-based volume renderer. In *IEEE International Symposium on Volume Visualization* (1998), pp. 39–46.

[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.

[CF07] Caniard F., Fleming R. W.: Distortion in 3D shape estimation with changes in illumination. In *Proceedings of the 4th Symposium on Applied Perception in Graphics and Visualization* (Tubingen, Germany, 2007), ACM, pp. 99–105.

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

[DEP05] Desgranges P., Engel K., Paladini G.: Gradient-free shading: A new method for realistic interactive volume rendering. In *International Fall Workshop on Vision, Modeling, and Visualization* (2005), pp. 209–216.

[DGMF11] Delalandre C., Gautron P., Marvie J.-E., François G.: Transmittance function mapping. In *Symposium on Interactive 3D Graphics and Games* (2011), pp. 31–38.

[DK09] Dachsbacher C., Kautz J.: Real-time global illumination. In *ACM SIGGRAPH Courses Program* (2009), pp. 2–36.

[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.

[DYV08] Díaz J., Yela H., Vázquez P.-P.: Vicinity occlusion maps—Enhanced depth perception of volumetric models. In *Computer Graphics International* (2008), pp. 56–63.

[GDM11] Gautron P., Delalandre C., Marvie J.-E.: Extinction transmittance maps. In *SIGGRAPH Asia 2011 Sketches* (2011), pp. 7:1–7:2.

[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.

[GRWS04] Geist R., Rasche K., Westall J., Schalkoff R.: Lattice-boltzmann lighting. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques* (Norrköping, Sweden, 2004), Eurographics Association, pp. 355–362.

[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.

[HKSB06] Hadwiger M., Kratz A., Sigg C., Bühler K.: GPU-accelerated deep shadow maps for direct volume rendering. In *ACM SIGGRAPH/EG Conference on Graphics Hardware* (2006), pp. 27–28.

[HLY07] Hernell F., Ljung P., Ynnerman A.: Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *IEEE/EG International Symposium on Volume Graphics* (2007), pp. 1–8.

[HLY08] Hernell F., Ljung P., Ynnerman A.: Interactive global light propagation in direct volume rendering using local piecewise integration. In *IEEE/EG International Symposium on Volume and Point-Based Graphics* (2008), pp. 105–112.

[HLY09] Hernell F., Ljung P., Ynnerman A.: Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 15*, *2* (2009), 548–559.

[HM03] Halle M., Meng J.: Lightkit: A lighting system for effective visualization. In *Visualization, 2003 (VIS 2003)* (Seattle, Washington, USA, 2003), IEEE, pp. 363–370.

[JB10] Jansen J., Bavoil L.: Fourier opacity mapping. In *SI3D: Symposium on Interactive 3D Graphics* (2010), pp. 165–172.

[Jen96] Jensen H. W.: Global illumination using photon maps. *Rendering Techniques* (1996), Springer-Verlag, Vienna, pp. 21–30.

[JGY*12] Jönsson D., Ganestam P., Ynnerman A., Doggett M., Ropinski T.: Explicit cache management for volume ray-casting on parallel architectures. In *EG Symposium on Parallel Graphics and Visualization (EGPGV)* (2012), pp. 31–40.

[JKRY12] Jönsson D., Kronander J., Ropinski T., Ynnerman A.: Historygrams: Enabling interactive global illumination in direct volume rendering using photon mapping. *IEEE Transactions on Visualization and Computer Graphics 18*, *12* (2012), 2364–2371.

[KD10] Kaplanyan A., Dachsbacher C.: Cascaded light propagation volumes for real-time indirect illumination. In *Symposium on Interactive 3D Graphics and Games (I3D'10)* (Washington, DC, USA, 2010), ACM, pp. 99–107.

[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.

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

[KPB12] Kroes T., Post F. H., Botha C. P.: Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE 7*, *7* (2012), e38586.

[KPH*03] Kniss J., Premoze S., Hansen C., Shirley P., McPherson A.: A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics 9*, *2* (2003), 150–162.

[KPHE02] Kniss J., Premoze S., Hansen C., Ebert D.: Interactive translucent volume rendering and procedural modeling. In *IEEE Visualization* (2002), pp. 109–116.

[KW03] Krüger J., Westermann R.: Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Visualization* (Seattle, Washington, USA, 2003).

[LB00] Langer M. S., Bülthoff H. H.: Depth discrimination from shading under diffuse lighting. *Perception 29*, *6* (2000), 649–660.

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

[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.

[LLY06] Ljung P., Lundström C., Ynnerman A.: Multiresolution interblock interpolation in direct volume rendering. In *IEEE/EG Symposium on Visualization* (2006), pp. 259–266.

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

[LR10] Lindemann F., Ropinski T.: Advanced light material interaction for direct volume rendering. In *IEEE/EG International Symposium on Volume Graphics* (2010), pp. 101–108.

[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.

[LV00] Lokovic T., Veach E.: Deep shadow maps. In *Proceedings of SIGGRAPH* (2000), pp. 385–392.

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

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

[MMC99] Mueller K., Möller T., Crawfis R.: Splatting without the blur. In *proceedings of the IEEE Visualization* (San Francisco, CA, USA, 1999), p. 61.

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

[MTK*01] Madison C., Thompson W., Kersten D., Shirley P., Smits B.: Use of interreflection and shadow for surface contact. *Perception & Psychophysics 63*, *2* (2001), 187–194.

[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 International Symposium on Volume Graphics* (2010), pp. 53–60.

[NM01] Nulkar M., Mueller K.: Splatting with shadows. In *Volume Graphics* (2001), pp. 35–50.

[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.

[PM08] Penner E., Mitchell R.: Isosurface ambient occlusion and soft shadows with filterable occlusion maps. In *IEEE/EG International Symposium on Volume and Point-Based Graphics* (2008), pp. 57–64.

[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.

[RBV*08] Ruiz M., Boada I., Viola I., Bruckner S., Feixas M., Sbert M.: Obscurance-based volume rendering framework. In *IEEE/EG International Symposium on Volume and Point-Based Graphics* (2008), pp. 113–120.

[RC06] Rodgman D., Chen M.: Refraction in volume graphics. *Graphical Models 68* (2006), 432–450.

[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.

[RDRS10b] Ropinski T., Döring C., Rezk-Salama C.: Interactive volumetric lighting simulating scattering and shadowing. In *IEEE Pacific Visualization* (2010), pp. 169–176.

[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.

[RKH08] Ropinski T., Kasten J., Hinrichs K. H.: Efficient shadows for GPU-based volume raycasting. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2008), pp. 17–24.

[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.

[RS07] Rezk-Salama C.: GPU-based Monte-Carlo volume raycasting. In *Proceedings of Pacific Graphics* (2007), pp. 411–414.

[RSHRL09] Rezk Salama C., Hadwiger M., Ropinski T., Ljung P.: Advanced illumination techniques for GPU volume raycasting. In *ACM SIGGRAPH Courses Program* (2009), pp. 1–166.

[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.

[SKS02] Sloan P., Kautz J., Snyder J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of SIGGRAPH* (2002), pp. 527–536.

[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 (Proceedings of SIGGRAPH)* (1992), pp. 249–252.

[SMG*13] Schott M., Martin T., Grosset A. P., Smith S. T., Hansen C. D.: Ambient occlusion effects for combined volumes and tubular geometry. *IEEE Transactions on Visualization and Computer Graphics 19*, *6* (2013), 913–926.

[SMGB12] Schott M., Martin T., Grosset A., Brownlee C.: Combined surface and volumetric occlusion shading. In *Proceedings of PacificVis* (2012), pp. 169–176.

[SMP11] Schlegel P., Makhinya M., Pajarola R.: Extinction-based shading and illumination in GPU volume ray-casting. *IEEE Transactions on Visualization and Computer Graphics 17*, *12* (2011), 1795–1802.

[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) 28*, *3* (2009), 855–862.

[SSKE05] Stegmaier S., Strengert M., Klein T., Ertl T.: A simple and flexible volume rendering framework for graphics-hardware-based raycasting. *International Workshop on Volume Graphics* (2005), 187–241.

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

[SYR11] Sundén E., Ynnerman A., Ropinski T.: Image plane sweep volume illumination. *IEEE Transactions on Visualization and Computer Graphics 17*, *12* (2011), 2125–2134.

[vPBV10] Šoltészová V., Patel D., Bruckner S., Viola I.: A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum (Eurographics/IEEE VGTC Symposium on Visualization 2010) 29*, *3* (2010), 883–891.

[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.

[vTPV12] Šoltészová V., Turkay C., Price M. C., Viola I.: A perceptual-statistics shading model. *IEEE Transactions on Visualization and Computer Graphics 18*, *12* (2012), 2265–2274.

[Wan92] Wanger L. C.: The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *SI3D: Symposium on Interactive 3D Graphics* (1992), pp. 39–42.

[WB08] Weigle C., Banks D.: A comparison of the perceptual benefits of linear perspective and physically-based illumination for display of dense 3D streamtubes. *IEEE Transactions on Visualization and Computer Graphics, 14*, *6* (2008), 1723–1730.

[WFG92] Wanger L. C., Ferwerda J. A., Greenberg D. P.: Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Application, 12*, *3* (1992), 44–58.

[WGLS05] Wang C., Gao J., Li L., Shen H.-W.: A multiresolution volume rendering framework for large-scale time-varying data visualization. In *International Workshop on Volume Graphics* (June 2005), pp. 11–223.

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

[WPSH06] Wyman C., Parker S., Shirley P., Hansen C.: Interactive display of isosurfaces with global illumination. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 186–196.

[ZC02] Zhang C., Crawfis R.: Volumetric shadows using splatting. In *IEEE Visualization* (2002), pp. 85–92.

[ZC03] Zhang C., Crawfis R.: Shadows and soft shadows with participating media using splatting. *IEEE Transactions on Visualization and Computer Graphics 9*, *2* (2003), 139–149.

[ZIK98] Zhukov S., Iones A., Kronin G.: An ambient light illumination model. In *EGRW: Proceedings of the Eurographics Workshop on Rendering* (1998), pp. 45–55.

[ZM13] Zhang Y., Ma K.-L.: Fast global illumination for interactive volume visualization. In *SI3D: Symposium on Interactive 3D Draphics* (2013), pp. 55–62.

[ZRL*08] Zhou K., Ren Z., Lin S., Bao H., Guo B., Shum H.-Y.: Real-time smoke rendering using compensated ray marching. *ACM Transactions on Graphics 27*, *3* (Aug. 2008), 36:1–36:12.

[ZXC05] Zhang C., Xue D., Crawfis R.: Light propagation for mixed polygonal and volumetric data. In *CGI: Proceedings of the Computer Graphics International* (2005), pp. 249–256.