Eurographics/ IEEE-VGTC Symposium on Visualization 2009
H.-C. Hege, I. Hotz, and T. Munzner
(Guest Editors)

Volume 28 (2009), Number 3

A Directional Occlusion Shading Model for Interactive Direct Volume Rendering

Mathias Schott[1], Vincent Pegoraro[1], Charles Hansen[1], Kévin Boulanger[2], Kadi Bouatouch[2]

[1]SCI Institute, University of Utah, USA [2]INRIA Rennes, Bretagne-Atlantique, France

Abstract

Volumetric rendering is widely used to examine 3D scalar fields from CT/MRI scanners and numerical simulation datasets. One key aspect of volumetric rendering is the ability to provide perceptual cues to aid in understanding structure contained in the data. While shading models that reproduce natural lighting conditions have been shown to better convey depth information and spatial relationships, they traditionally require considerable (pre)computation. In this paper, a shading model for interactive direct volume rendering is proposed that provides perceptual cues similar to those of ambient occlusion, for both solid and transparent surface-like features. An image space occlusion factor is derived from the radiative transport equation based on a specialized phase function. The method does not rely on any precomputation and thus allows for interactive explorations of volumetric data sets via on-the-fly editing of the shading model parameters or (multi-dimensional) transfer functions while modifications to the volume via clipping planes are incorporated into the resulting occlusion-based shading.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism— Subjects: Color, shading, shadowing, and texture

1. Introduction

Volumetric rendering is widely used to examine 3D scalar fields from CT/MRI scanners and numerical simulation datasets. One key aspect of volume rendering is the ability to provide shading cues to aid in understanding structures contained in the datasets. Various shading models have been proposed to improve the comprehensibility of complex features in volumes. Especially methods which take the relative spatial arrangement of features into account have been successfully applied to enhance the quality of visualizations of volumetric data sets.

An example of such techniques is ambient occlusion (AO), defined as the amount of light reaching a point from a spherical environment light source with uniform intensity enclosing the whole scene. The surrounding area light source causes occluding structures to cast soft shadows, giving critical cues to a human observer about their relative spatial arrangement [LB00]. Computing AO is very challenging, since the required global information about the volume makes the task computationally intensive and commonly involves precomputation as an approach to make the technique feasible. Expensive computation however conflicts with the desire for an interactive volume rendering technique with user flexibility. Allowing the change of transfer functions or clipping planes precludes expensive precomputations and makes it necessary to recompute shading parameters for every frame.

This paper presents an occlusion-based shading method yielding depth cues similar to those of AO. The proposed method extends [KPH*03] to render occlusion effects interactively and thus allows user driven-manipulation of multi-dimensional transfer functions, which have been shown to improve the classification of materials in data sets of complex composition [KKH02]. Unlike in precomputed AO methods, clipping planes influence the occlusion effects. Transparent and solid surfaces are shaded to give additional insight when it is desirable to show multiple occluding surfaces at the same time, and volumetric data sets with high feature complexity benefit from the presented method.

This paper is structured as follows: Section 2 discusses related work focusing on volumetric shading. Section 3 derives directional occlusion shading from the physics of radiative transport and outlines an integration into a slice-based volume renderer. Results are presented and discussed in Sec-

tion 4, followed by conclusions and future work in Section 5.

2. Related Work

The shading model used during volume rendering plays a critical role in helping the user gain insight during the exploration of a volumetric data set. Simple approximations to light transport, such as emission-absorption models [Max95] or local illumination shading models such as the Blinn illumination model [Bli77] build the foundation of many volume rendering systems, thanks to their low computational cost and simplicity of implementation.

However, light contributions from surrounding features give important perceptual cues supporting the comprehension and understanding of complex features in volumetric data sets. Shadows have been incorporated into volume rendering systems, as basic means of gaining additional lighting and occlusion information, e.g. as precomputed shadow volumes [BR98], variations of image space shadow maps [KKH02, DEP05, HKSB06] or via ray-casting [RKH08].

AO methods, especially for polygonal geometry, have recently received considerable attention since they provide, with an expensive preprocessing step, a method to realistically shade scenes under uniform diffuse illumination, at a cheaper cost than full global illumination [LB00]. Here, we will only discuss AO techniques for volume rendering and refer the reader to a recent survey [MFS09] on AO techniques in general.

The Vicinity Shading [Ste03] and Obscurance Shading [RBV*08] methods determine as a preprocessing step a vicinity/obscurance value for each voxel by taking surrounding voxels into consideration. Occlusion is based on voxel densities for Vicinity Shading, or on the distance between occluding voxels for Obscurance Shading, which also supports color bleeding. Both methods require a recomputation of the occlusion values when volume classification parameters such as the iso-value or the transfer function change.

Vicinity Occlusion Maps [DYV08] compute image space halos by comparing depth values with the surrounding average depth values as a measure for the arrangement of voxels in the vicinity. Volumetric halos are also used to enhance depth perception in illustrative volume rendering [BG07].

Hernell *et al.* [HLY07] compute a local AO approximation by considering voxels in a neighborhood around each voxel under illumination of a spherical light source. Ray-casting is used to compute local ambient light and emmissive contributions, which they store as volume textures using multi-resolution block-based data structures and spatial under sampling. Changing the transfer function causes an interactive incremental refinement of the ambient and emissive illumination volumes as well as a recomputation of the transfer function depending block-based data structures. A subsequent publication [HLY08] extends this local AO shading with global shadows and first order scattering effects.

Ritschel [Rit07] considers the illumination of a volume data set under distant spherical illumination represented by an environment map. A visibility function is computed by integrating the transmittance along hundreds of rays emanating from each voxel using spatial undersampling and an adaptive traversal of a hierarchical data structure. A spherical harmonics decomposition of both the visibility function and the environment maps allows efficient storage and evaluation of low frequency indirect illumination values during direct volume rendering. Changing the transfer function requires a few seconds for recomputation of the visibility function. Rezk-Salama *et al.* [RS07] use GPUs to accelerate Monte-Carlo volume ray-casting to render isosurfaces with high-quality scattering, AO and refraction effects.

Desgrange *et al.* [DE07] use a linear combination of opacity volumes blurred with different filter sizes to approximate the occlusion by the average opacity of surrounding voxels. They use summed area volumes to compute the average opacity volumes after transfer function changes.

Filterable occlusion maps [PM08] use an approach similar to variance shadow maps [DL06] to represent statistical information about neighboring voxels which are used to compute AO and soft shadow effects during the rendering of isosurfaces. The computation of the filterable occlusion maps runs interactively on recent GPUs and arbitrary iso-values can be chosen, but inaccuracies appear if the statistical approximation is not representative of the actual distribution of the dataset. Filterable occlusion maps can be combined with direct volume rendering, but semi-transparent materials don't influence the occlusion computation.

Ropinski *et al.* [RMSD*08] compute and quantize the configuration of neighboring voxels independently of a specific transfer function as a lengthy precomputation step. Combining those precomputed values with a transfer function allows them to interactively render AO and color bleeding effects. In contrast to their method, we support multi-dimensional transfer functions and post-classification.

The method proposed by Kniss *et al.* [KPH*03] uses direct and indirect light contributions to render translucency effects, as demonstrated on clouds and regions of semi-dense materials, such as a block of wax. The indirect light contribution is evaluated using successive blurring of an additional buffer, where chromatic attenuation of lower magnitude than that of the achromatic direct light contribution is used to let the light penetrate deeply into the material, qualitatively approximating scattering effects of materials with a strongly forward peaked phase function.

The proposed directional occlusion shading model uses an additional buffer to compute additional shading model parameters similar to Kniss *et al.* [KPH*03], but differs by using it to accentuate semi-transparent solid and surface-like features via a viewer-oriented backward-peaked cone phase function, instead of approximating translucency effects of semi-dense materials with strongly forward scatter-

ing phase functions. It supports perspective correct sampling and attenuation of the occlusion buffer and an implementation exploiting multiple render targets to reduce CPU and GPU costs of the evaluation of the presented shading model. Floating point buffers are used to increase the precision, especially for low-transparency materials and low inter-slice distances.

3. Directional Occlusion Shading

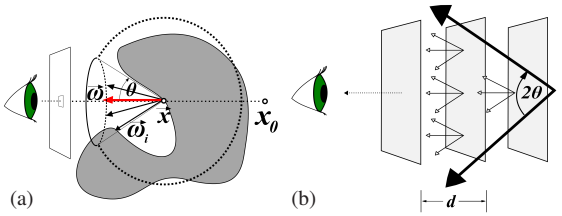

Figure 1: *a) The geometric setup of the proposed direct occlusion shading model showing the conical subset of the sphere used to compute the occlusion. b) The setup used for the interactive computation of the occlusion factors.*

The proposed directional occlusion shading (DOS) model captures the important perceptional cues caused by soft shadow effects of AO techniques without having to evaluate a prohibitively expensive global AO solution. Unlike previous approaches which only consider local occlusion in a spherical neighborhood, DOS takes into account all features between a point and the ambient light. However, instead of computing occlusion on the whole sphere, a specialized phase function, namely a backward-peaked cone phase function of user-specified aperture angle as illustrated in Figure 1(a), is used to compute occlusion effects. This is in spirit similar to the forward-peaked *light transport angle* used to render scattering effects interactively [KPH*03] or to the *cone angle* used by artists to control AO effects in offline rendering [Chr03].

A Monte-Carlo raytracer was used to empirically compare directional occlusion (Figure 2(b)) and volumetric ambient occlusion with an isotropic phase function (Figure 2(a)). While the results differ, the cone phase function is able to highlight features of interest for visualization purposes in a manner similar to full ambient occlusion. Although the interactive approximation (Figure 2(c)) presents slight differences with the reference images, it allows most of the complex shading effects to be rendered at interactive frame rates without any precomputation.

By changing the aperture angle of the cone, as discussed in Section 4, it is possible to vary the influence of features at different distances, thus allowing the user to choose a balance between visualizing local and global structures. A complete evaluation of the occlusion, albeit being closer to the physics of light transport, would restrict this flexibility.

This section outlines the radiative transport equation and the derivation of the directional occlusion shading model. Given this theoretical framework, it is shown how to extend a slice-based 3D texture volume renderer to support DOS.

3.1. Radiative Transport Equation

The radiance integrated along a ray with origin $\vec{x_0}$ and direction $\vec{\omega}$, passing through a participating medium is described by the radiative transport equation (RTE) [CPCP*05]:

$$L(\vec{x},\vec{\omega}) = T(\vec{x},\vec{x_0})L_b(\vec{x_0},\vec{\omega}) + L_m(\vec{x},\vec{\omega}) \quad (1)$$

$$L_m(\vec{x},\vec{\omega}) = \int_{\vec{x}}^{\vec{x_0}} T(\vec{x},\vec{x'})\sigma_a(\vec{x'})L_e(\vec{x'},\vec{\omega})d\vec{x'} + \int_{\vec{x}}^{\vec{x_0}} T(\vec{x},\vec{x'})\sigma_s(\vec{x'})L_i(\vec{x'},\vec{\omega})d\vec{x'} \quad (2)$$

This equation expresses the radiance at a given point as the sum of the attenuated background radiance $L_b(\vec{x_0},\vec{\omega})$ and the medium radiance $L_m(\vec{x},\vec{\omega})$. The latter integrates the contributions of the emitted radiance $L_e(\vec{x},\vec{\omega})$, weighted by the wavelength-dependent absorption coefficient σ_a, and the in-scattered radiance $L_i(\vec{x},\vec{\omega})$, also weighted by the wavelength-dependent scattering coefficient σ_s. The extinction coefficient $\sigma_t = \sigma_a + \sigma_s$ describes the amount of radiance lost through absorption and out-scattering via the transmittance $T(\vec{x_a},\vec{x_b})$ and the optical thickness $\tau(\vec{x_a},\vec{x_b})$.

$$T(\vec{x_a},\vec{x_b}) = e^{-\tau(\vec{x_a},\vec{x_b})} \quad (3)$$

$$\tau(\vec{x_a},\vec{x_b}) = \int_{\vec{x_a}}^{\vec{x_b}} \sigma_t(\vec{x})d\vec{x} \quad (4)$$

The emitted radiance only depends on local properties of the medium, but the in-scattered radiance $L_i(\vec{x'},\vec{\omega})$ depends on global properties, since it is defined as the integral of the radiance incoming at the point $\vec{x'}$ over all directions $\vec{\omega_i}$, weighted by the phase function $\Phi(\vec{\omega},\vec{\omega_i})$.

$$L_i(\vec{x'},\vec{\omega}) = \int_{4\pi} L(\vec{x'},\vec{\omega_i})\Phi(\vec{\omega},\vec{\omega_i})d\vec{\omega_i}. \quad (5)$$

The recursive evaluation of $L(\vec{x'},\vec{\omega_i})$ in Equation 5 makes the RTE in its generality very challenging to evaluate. However, simplified models suffice for visualization purposes while allowing for a significant reduction in computational cost. The next section will describe such a specialized model.

3.2. Directional Occlusion Shading Model

The directionality of the occlusion evaluation is modeled with a cone-shaped phase function $\Phi_\theta(\vec{\omega},\vec{\omega_i})$, as given in Equation 6. The aperture angle of the cone is $\theta \in [0,\frac{\pi}{2})$.

$$\Phi_\theta(\vec{\omega},\vec{\omega_i}) = \begin{cases} \frac{1}{2\pi(1-\cos\theta)} & \text{if} \langle\vec{\omega},\vec{\omega_i}\rangle > \cos(\theta) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The radius $r = \frac{1}{2\pi(1-\cos\theta)}$ normalizes the phase function to $\int_{4\pi}\Phi(\vec{\omega},\vec{\omega_i})d\vec{\omega_i} = 1$. In this model, $\vec{\omega}$ corresponds to the

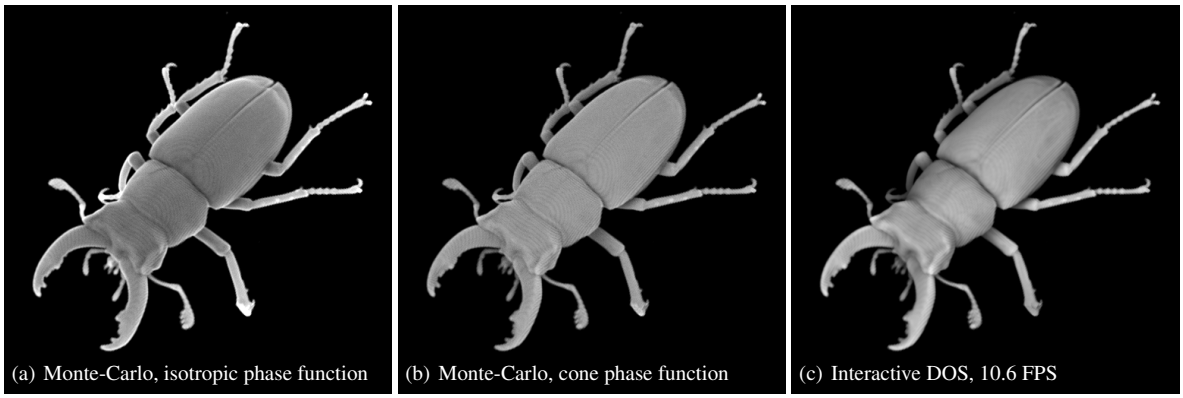

Figure 2: *The stag beetle data set rendered using a) Monte-Carlo integration of Equation 7, with an isotropic phase function b) Monte-Carlo integration of Equation 7, with a cone phase function of aperture angle* $\theta = 80^\circ$ *and c) interactive directional occlusion shading with 1046 slices and an aperture of* 80°.

viewing direction and the $\vec{\omega}_i$ are chosen to be within the cone such that $\Phi(\vec{\omega},\vec{\omega}_i) \neq 0$. Then, the RTE is simplified by assuming that the medium does not emit light ($L_e(\vec{x'},\vec{\omega}) = 0$) and scatters light only at directions within a cone, described by the phase function $\Phi_\theta(\vec{\omega},\vec{\omega}_i)$. Only first-order scattering events are considered for the in-scattered radiance. For a certain ray with direction $\vec{\omega}_i$, leaving the volume at $x_{0,i}^{\vec{}}$, $L(\vec{x'},\vec{\omega}_i)$ equals to the constant ambient radiance L_a attenuated by the transmittance $T(\vec{x'},x_{0,i}^{\vec{}})$ along $\vec{\omega}_i$ and scaled by $\Phi_\theta(\vec{\omega},\vec{\omega}_i)$; dropping all other terms of the recursive evaluation of $L(\vec{x},\vec{\omega})$ in Equation 5. Equations 7 to 10 describe the model for DOS:

$$L(\vec{x},\vec{\omega}) = T(\vec{x},\vec{x_0})L_b(\vec{x_0},\vec{\omega}) + L_m(\vec{x},\vec{\omega}) \quad (7)$$

$$L_m(\vec{x},\vec{\omega}) = \int_{\vec{x}}^{\vec{x_0}} T(\vec{x},\vec{x'})\sigma_s(\vec{x'})L_i(\vec{x'},\vec{\omega})d\vec{x'} \quad (8)$$

$$L_i(\vec{x'},\vec{\omega}) = L_a V(\vec{x'},\vec{\omega}) \quad (9)$$

$$V(\vec{x'},\vec{\omega}) = \int_{4\pi} T(\vec{x'},x_{0,i}^{\vec{}})\Phi_\theta(\vec{\omega},\vec{\omega}_i)d\vec{\omega}_i \quad (10)$$

The in-scattered radiance $L_i(\vec{x'},\vec{\omega})$ in Equation 9 can be further decomposed into the product of a fractional visibility term $V(\vec{x'},\vec{\omega})$, as given in Equation 10, and the ambient radiance L_a. The fractional visibility is conceptually equivalent to the *occlusion factor* of surface AO methods, since it describes how much of the environment's light is reaching the point $\vec{x'}$. Section 3.4 explains how to use additional buffers to interactively calculate occlusion factors using a slice-based volume renderer.

It is difficult to measure σ_s and σ_t for real world data-sets, since typical CT/MRI scanners measure densities or orientations of magnetic fields. Therefore, 2D transfer functions are used to map attributes specified at positions in the volume to the chromatic scattering coefficient σ_s and the achromatic extinction coefficient σ_t, dropping σ_a, since $L_e(\vec{x'},\vec{\omega}) = 0$.

3.3. Rendering Algorithm

The proposed algorithm, as outlined in Algorithms 1 and 2, can be integrated into a slice-based volume renderer [WVW94]. Graphics hardware is used to render proxy geometry, typically slices created by intersecting view aligned planes with the bounding box of the volume, using a fragment program to compute color and opacity values to be composited with the image of previous slices already stored in the frame buffer.

Similar to half-angle based methods [KKH02, KPH*03], an additional buffer is used to store and compute the occlusion factor for each slice. The eye buffer and the occlusion buffer get updated alternately during the slice-by-slice traversal of the volume. An additional render pass updates the occlusion buffer, approximating Equation 10 by averaging samples taken from the previous occlusion buffer.

After computing the world space proxy geometry on the CPU, view-aligned slices are rendered at first from the eye's point of view, projecting them into the eye buffer. During this pass, the occlusion factor is determined by reading the occlusion buffer at the projected fragment position, multiplying it by the color of the sample and the ambient radiance, blending into the eye buffer using front-to-back compositing.

Next, the occlusion information is updated by rendering the slice again, but now into the next occlusion buffer. During this pass, the occlusion factors of the previous occlusion buffer are integrated by reading and averaging multiple samples, as described in greater detail in Section 3.4. Finally, the current and next occlusion buffers are swapped and the algorithm proceeds to the next slice.

When the final eye buffer has been computed, it gets copied into the frame buffer of the visible window, converting the final values from floating-point format to the fixed-point format usually used for displaying images on the

screen. If required, tone-mapping can be applied here.

3.4. Interactive Computation of the Occlusion Factor

It is possible to approximate the integration of the occlusion factors described in Equation 10 by averaging attenuated samples from the previous occlusion buffer in a neighborhood around the current fragment for propagation of the occlusion to the current occlusion buffer. Figure 1(b) shows the setup for the computation of the occlusion factors. Given the aperture angle of the cone θ and the inter-slice distance d, the *occlusion_extent* is defined as the radius of the cone's base enclosing all samples, and computed by Equation 11.

$$occlusion\_extent = d\tan(\theta) \tag{11}$$

The use of view-aligned slices causes the cone axis to be collinear with the slice normal. However, perspective projection requires a correction factor to compensate for the different size of the base of the cone in screen space since the *occlusion_extent* varies as a function of the z coordinate of the slice. A correction factor $\vec{c}$, as described in Equation 12, is computed in the vertex shader using the projection matrix P and the view space z-component of the vertices of the slices, which is constant across a given slice. The correction factor $\vec{c}$ is then passed to the fragment shader where it is used to compute texture coordinates for sampling the previous occlusion buffer, given in Equation 15.

$$\vec{c} = \begin{pmatrix} \frac{x_{scale}}{w_{scale}} \\ \\ \frac{y_{scale}}{w_{scale}} \end{pmatrix} \qquad \begin{pmatrix} x_{scale} \\ y_{scale} \\ z_{scale} \\ w_{scale} \end{pmatrix} = P \begin{pmatrix} 1 \\ 1 \\ z \\ 1 \end{pmatrix} \tag{12}$$

Sample positions $\vec{p}$ are generated on the disk with radius *occlusion_extent* and transformed according to Equations 13 to 15. A perspective division projects the interpolated clip space position $x\vec{y}_c$ of a fragment into normalized device coordinates $x\vec{y}_d$. Then, a sample position $\vec{p}$ on a disk with radius *occlusion_extent* is scaled component-wise (here denoted by $\otimes$) by the correction factor $\vec{c}$ and added to $x\vec{y}_d$. This yields a sample position in normalized device coordinates $\vec{p}_d$ which is scaled and biased by 0.5 to determine the texture coordinates $\vec{p}_t$ to sample the previous occlusion buffer.

$$x\vec{y}_d = \frac{x\vec{y}_c}{w_c} \tag{13}$$

$$\vec{p}_d = x\vec{y}_d + \vec{c} \otimes \vec{p} \tag{14}$$

$$\vec{p}_t = 0.5 * \vec{p}_d + \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \tag{15}$$

Computing the integral approximation is done by reading the previous occlusion buffer at all sample positions $\vec{p}_t$, attenuating each sample by the transmittance computed using the distance between the tip of the cone and the sample position, to finally average the weighted samples.

To generate sample positions p, a uniform grid of user specifiable resolution has been chosen as a compromise between performance and image quality. Its symmetry and orientation of samples along the axes of the occlusion buffer reduce inter-frame continuity artifacts when the camera position changes.

4. Results and Discussion

The methods was implemented using OpenGL and Cg running on an NVIDIA GeForce GTX 280 GPU with 1 GB. The images were rendered at a 512×512 resolution into 16-bit precision floating-point buffers with an inter-slice distance of $d = 0.001$ to capture high-frequency details introduced by multi-dimensional transfer functions.

Multiple render targets (MRT) allow a parallel update of both eye and occlusion buffers. Rendering into/reading from individual layers of the same layered render target and performing blending operations manually in the fragment program reduce the CPU and GPU overhead of the described single render target (SRT) implementation. This exploits behavior undefined by the OpenGL specification, but works in practice, since it ensures that reads and writes do not reference the same layer. MRT performance increases between 44% and 99%, compared to the presented SRT implementation, especially for low number of occlusion buffer samples. For higher number of samples, the bottleneck shifts toward memory bandwidth, thus reducing the gain from exploiting MRTs. The reported frame rates are from the MRT implementation.

Experimentation showed no qualitative visual difference between using the exact distance between the tip of the cone and a sample point, $distance = \sqrt{d^2 + \langle \vec{p}, \vec{p} \rangle}$, during computation of $T(\vec{x'}, \vec{x_{0,i}})$, or approximating it with the inter-slice distance d, as outlined in Algorithm 3. This increased the performance up to 23% (between 30% to 98% if also using MRTs) for relatively high number of samples. The presented images have all been rendered using this approximation.

Figure 3 shows the directional occlusion shading model and the Lambertian diffuse shading model applied to different data sets. Even though the diffuse shading model provides basic hints about shape, the directional occlusion shading model accentuates structures such as concavities and depth discontinuities. The creases of the brain in Figure 3(a) and the eye sockets in Figure 3(b) are examples of the former; the soft shadows cast by the zygomatic bone onto the sphenoid bone in Figure 3(b) and the emphasized boundaries of the fish bones with respect to the background in Figure 3(c) illustrate the latter.

Figure 4(a) shows occlusion of transparent skin, visible at the wrinkles as well as the nose accentuating the solid bone surface below. In Figure 4(b), a clipping plane has been used to uncover otherwise hidden details of an engine block. Intricate details of a tree data set are emphasized in Figure 4(c),

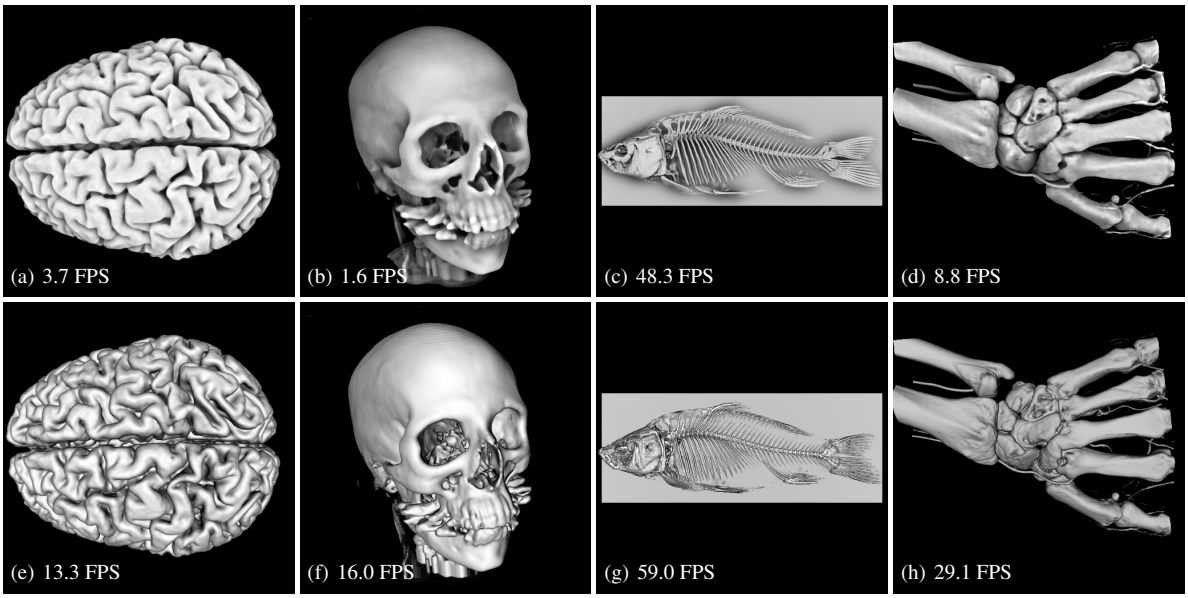

Figure 3: *Various data sets showing the difference between directional occlusion shading (top row) and diffuse Lambertian shading (bottom row) for different data sets. Two-dimensional transfer functions have been used to highlight different features of an MRI scan of a brain with a resolution of* $256 \times 256 \times 160$ *voxels, a CT scan of a head with a resolution of* $128 \times 256 \times 256$ *voxels, a CT scan of a carp with a resolution of* $256 \times 288 \times 512$ *voxels and a CT scan of a hand with a resolution of* $244 \times 124 \times 257$ *voxels. The number of slices used to render each image are 1000, 1458, 220 and 619 respectively.*

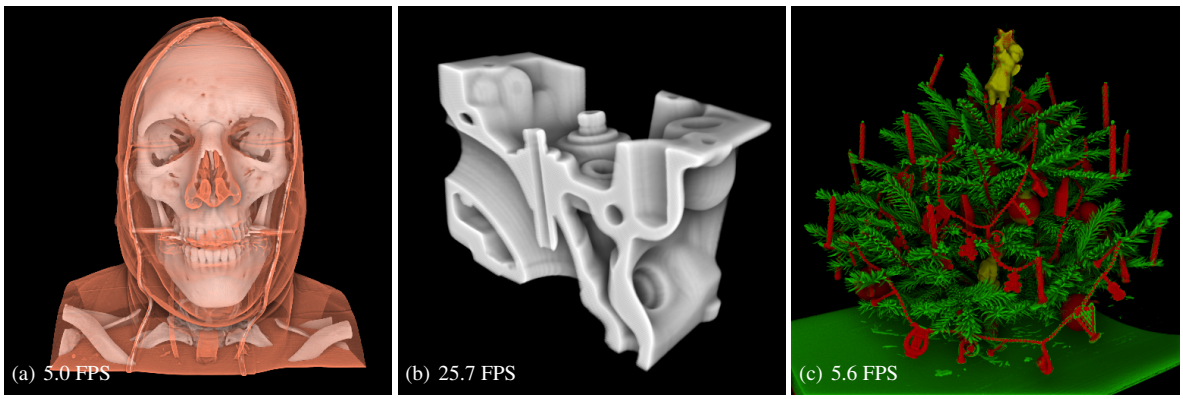

Figure 4: *a) Visible male data set with occlusion of solid and transparent materials (*$512 \times 512 \times 302$ *voxels, 962 slices) b) CT scan of an engine block where a clipping plane was used to show the exhaust port (*$256 \times 256 \times 128$ *voxels, 679 slices) c) Tree data set of which complex features are exposed by the ambient occlusion approximation (*$512 \times 512 \times 512$ *voxels, 1563 slices)*

where 2D transfer functions were used to properly classify different materials contained in the decoration and the tree.

Figures 3(a) and 3(b) show regular structures, suggesting rendering artifacts of the proposed shading model, especially since those structures are not as visible with diffuse shading (Figures 3(e) and 3(f)). For diffuse shading, the gradient is precomputed and stored in an additional 3D texture with the same resolution as the volume, and accessed during shading by a trilinearly interpolated texture fetch. This process implicitly applies a low pass filter to the gradients, the key shading parameter of diffuse shading, and ignores high-frequency details introduced by the application of the transfer function. Directional occlusion shading on the other hand computes and stores shading parameters in image space at higher resolution, while capturing and emphasizing high frequency details introduced by the data interpolation and classification, thus accentuating the discretized structure

of a volumetric dataset. In the case of diffuse shading, on-the-fly gradient estimation of the classified data yields similar rendering artifacts which can be reduced, e.g. by using higher order filtering or by using data sets of higher resolution. On-the-fly gradient estimation trades memory required to store precomputed gradients for considerably reduced performance, due to multiple dependent texture fetches.

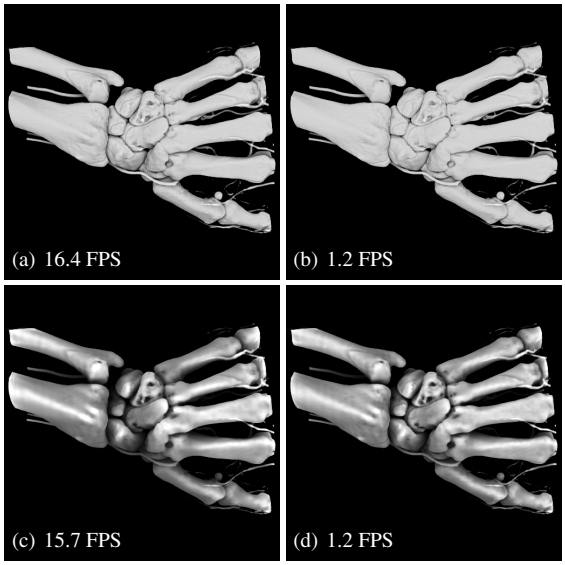

Figure 5: *The value of* θ *(top row:* $\theta = 50^\circ$*, bottom row:* $\theta = 85^\circ$*) and the resolution of the sampling grid (left column:* 2×2*, right column:* 13×13*) determine the effect of surrounding features. In the two images at the top, the base of the cone covers between 0.8 and 1.3 texels for a* 512×512 *view port. In the two images at the bottom, the base of the cone covers between 8.5 and 12.7 texels, for the same view port resolution. The images were rendered with 619 slices.*

Figure 5 shows the effect of varying the aperture angle of the cone phase function. Higher values of θ cause more distant features to increase the shading variation of surfaces, as demonstrated in Figures 5(c) and 5(a) which contrast a value of $\theta = 50^\circ$ with a value of $\theta = 85^\circ$. Ambient occlusion and obscurance-based techniques expose similar levels of control by letting the length of the occlusion rays determine the influence of more distant features in the volume.

Increasing the resolution of the sampling grid reduces the aliasing especially for high values of θ where the base of the cone covers a relatively high numbers of texels. For example in Figures 5(c) and 5(d), the cone aperture angle is $\theta = 85^\circ$, thus the occlusion extent varies between 8.5 and 12.7 units in texel space, depending on the z value of the slice. For low values of θ, such as in Figures 5(a) and 5(b), the occlusion extent covers only a relatively small region of the occlusion buffer, therefore there are only small differences between the different grid resolutions. However, the performance is strongly dependent on the grid resolution, as one can see in the reported frame rates.

5. Conclusion and Future Work

In this paper, a shading model has been presented which adds occlusion effects to direct volume rendering. Restricting the occlusion computation to a cone oriented toward the viewer enables interactive rendering of plausible occlusion effects, qualitatively similar to those of full ambient occlusion techniques. The method does not rely on pre-computation and therefore allows interactive manipulation of camera position, transfer function and clipping planes, which are all taken into account during the occlusion computation.

In the future, we would like to extend our shading model to render volumetric soft shadows from arbitrary area light sources by adapting half-angle based volume shading techniques. Integration of DOS into volume rendering methods based on ray-casting would be also desirable, as well as the application to rendering of iso-surfaces.

Acknowledgements

We would like to thank the Computer Graphics Groups of the Universities of Vienna, Erlangen and Sylvain Gouttard, University of Utah, for providing the datasets and reviewers for their comments and suggestions. This work has been funded by grants from the NSF, DOE and INRIA.

References

[BG07] BRUCKNER S., GRÖLLER E.: Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1344–1351.

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 11*, 2 (1977), 192–198.

[BR98] BEHRENS U., RATERING R.: Adding shadows to a texture-based volume renderer. In *Proceedings of the IEEE Symposium on Volume Visualization* (1998), pp. 39–46.

[Chr03] CHRISTENSEN P. H.: Global illumination and all that. *SIGGRAPH course notes 9 (RenderMan: Theory and Practice)* (2003).

[CPCP*05] CEREZO E., PEREZ-CAZORLA F., PUEYO X., SERON F., SILLION F.: A survey on participating media rendering techniques. *The Visual Computer 21*, 5 (2005), 303–328.

[DE07] DESGRANGES P., ENGEL K.: Fast ambient occlusion for direct volume rendering. *United States Patent Application*, #20070013696 (2007).

[DEP05] DESGRANGES P., ENGEL K., PALADINI G.: Gradient-free shading: A new method for realistic interactive volume rendering. *Proceedings of Vision, Modeling, and Visualization* (2005), 209–216.

[DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (2006), pp. 161–165.

[DYV08] DÍAZ J., YELA H., VÁZQUEZ P.: Vicinity occlusion maps: Enhanced depth perception of volumetric models. In *Computer Graphics International* (2008).

[HKSB06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K.: GPU-accelerated deep shadow maps for direct volume rendering. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware* (2006), pp. 49–52.

[HLY07] HERNELL F., LJUNG P., YNNERMAN A.: Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *IEEE/EG 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 Symposium on Volume and Point-Based Graphics* (2008), pp. 105–112.

[KKH02] KNISS J., KINDLMANN G., HANSEN C.: Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 270–285.

[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.

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

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

[MFS09] MÉNDEZ-FELIU A., SBERT M.: From obscurances to ambient occlusion: A survey. *The Visual Computer 25*, 2 (2009), 181–196.

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

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

[Rit07] RITSCHEL T.: Fast GPU-based visibility computation for natural illumination of volume data sets . In *Eurographics Short Papers* (2007), pp. 57–60.

[RKH08] ROPINSKI T., KASTEN J., HINRICHS K. H.: Efficient shadows for GPU-based volume raycasting. In *Proceedings of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)* (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 (Proceedings of Eurographics) 27*, 2 (2008), 567–576.

[RS07] REZK-SALAMA C.: Gpu-based monte-carlo volume raycasting. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (2007), pp. 411–414.

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

[WVW94] WILSON O., VANGELDER A., WILHELMS J.: *Direct volume rendering via 3D textures*. Tech. rep., University of California at Santa Cruz, Santa Cruz, CA, USA, 1994.

Appendix A: Pseudocode for the rendering algorithms

Algorithm 1 The main rendering algorithm (SRT)

ComputeProxyGeometry
$occlusion\_buffer_{prev} \Leftarrow occlusion\_buffer_{next} \Leftarrow 1$
$eye\_buffer \Leftarrow L_b(\vec{x_0}, \vec{\omega})$
for all slices s of proxy geometry **do**
  BindTexture($occlusion\_buffer_{prev}$)
  SetRenderTarget(eye_buffer)
  SetBlendingMode($1 - destination_\alpha, 1$)
  for all fragments f of s **do**
    $(x, |\nabla x|) \Leftarrow$ Texture3D($volume$,f)
    $(\sigma_s, \sigma_t) \Leftarrow$ Texture2D($transfer\_function$, $(x, |\nabla x|)$)
    $occlusion \Leftarrow$ Texture2D($occlusion\_buffer_{prev}$, f_{clip_space})
    $\alpha \Leftarrow 1 - e^{-\sigma_t * slice\_distance}$
    $frame\_buffer \Leftarrow (L_a * \sigma_s * occlusion * \alpha, \alpha)$
  end for
  BindTexture($occlusion\_buffer_{previous}$)
  SetRenderTarget($occlusion\_buffer_{next}$)
  DisableBlending()
  compute $\vec{c}$ in the vertex shader as in Equation 12
  for all fragments f of s **do**
    $(x, |\nabla x|) \Leftarrow$ Texture3D($volume$,f)
    $(\sigma_t) \Leftarrow$Texture2D($transfer\_function$, $(x, |\nabla x|)$)
    $occlusion \Leftarrow$ Run(Algorithm 2 or Algorithm 3)
    $frame\_buffer \Leftarrow occlusion$
  end for
  Swap($occlusion\_buffer_{prev}$, $occlusion\_buffer_{next}$)
end for
CompositWithWindowFrameBuffer(eye_buffer)

Algorithm 2 occlusion using correct distances

$occlusion \Leftarrow 0$
for all samples $\vec{p}$ on the base of the cone **do**
  compute $\vec{p_t}$ as in Equation 15
  $distance \Leftarrow \sqrt{slice\_distance^2 + \langle \vec{p}, \vec{p} \rangle}$
  $transmittance \Leftarrow e^{-\sigma_t * distance}$
  $sample \Leftarrow$ Texture2D($occlusion\_buffer_{prev}$, $\vec{p_t}$)
  $occlusion += sample * transmittance$
end for
$occlusion \Leftarrow occlusion/|p|$

Algorithm 3 occlusion using approximated distance

$occlusion \Leftarrow 0$
for all samples $\vec{p}$ on the base of the cone **do**
  compute $\vec{p_t}$ as in Equation 15
  $occlusion +=$ Texture2D($occlusion\_buffer_{prev}$, $\vec{p_t}$)
end for
$occlusion \Leftarrow (occlusion/|p|)e^{-\sigma_t * slice\_distance}$