Preliminary Version.
The definite version will be available at diglib.eg.org
and www.blackwell-synergy.com.

Interactive Volume Rendering with Dynamic Ambient Occlusion and Color Bleeding

Timo Ropinski, Jennis Meyer-Spradow, Stefan Diepenbrock, Jörg Mensmann and Klaus Hinrichs

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

Abstract

We propose a method for rendering volumetric data sets at interactive frame rates while supporting dynamic ambient occlusion as well as an approximation to color bleeding. In contrast to ambient occlusion approaches for polygonal data, techniques for volumetric data sets have to face additional challenges, since by changing rendering parameters, such as the transfer function or the thresholding, the structure of the data set and thus the light interactions may vary drastically. Therefore, during a preprocessing step which is independent of the rendering parameters we capture light interactions for all combinations of structures extractable from a volumetric data set. In order to compute the light interactions between the different structures, we combine this preprocessed information during rendering based on the rendering parameters defined interactively by the user. Thus our method supports interactive exploration of a volumetric data set but still gives the user control over the most important rendering parameters. For instance, if the user alters the transfer function to extract different structures from a volumetric data set the light interactions between the extracted structures are captured in the rendering while still allowing interactive frame rates. Compared to known local illumination models for volume rendering our method does not introduce any substantial rendering overhead and can be integrated easily into existing volume rendering applications. In this paper we will explain our approach, discuss the implications for interactive volume rendering and present the achieved results.

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

1. Introduction

The illumination models predominantly used for interactive volume rendering are only capable of representing local illumination phenomena. However, diffuse interreflections resulting in color bleeding are the predominant illumination characteristics influencing the perception of natural objects [AHH*94]. Furthermore it has been shown that these subtle effects support spatial comprehension and even improve perception of shapes compared to direct lighting [LB00]. Hence a sophisticated illumination model for volume rendering should take into account not only strictly local illumination phenomena, which are already covered by the Blinn-Phong model, but also those phenomena which depend on neighboring structures, such as color bleeding. However, since the neighboring scene elements have to be taken into account, simulating this phenomenon is computationally expensive, not allowing high frame rates which are required for interactive exploration.

In the past many methods based on the theory of light transfer have been developed to enable interactive diffuse interreflections for polygonal models [SHHS03, CHH03]. Most of these physically motivated approaches are based on pre-computing illumination for all vertices and storing it in an appropriate data structure which is accessed during rendering. Thus these algorithms support interactive modification of light and camera parameters as well as some material parameters. However, the application to deformable geometry is constrained and requires a new pre-computation in most cases [SLS05]. To address these limitations, approximations have been proposed, which are not physically moti-

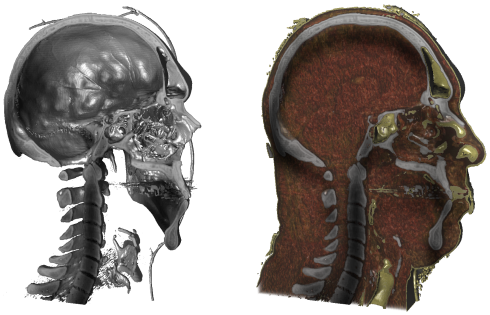


Figure 1: *The Visible Human head data set (192 × 192 × 110 voxel) has been clipped and rendered with our interactive volume rendering method. The two representations differ only in the transfer function which can be changed interactively. Although the visualized structures vary tremendously, light interactions between them become visible.*

vated, but lead to visually convincing results. In contrast to polygonal models the structure represented by a volumetric data set depends on the rendering parameters, which include the transfer function as well as thresholding parameters, since these rendering parameters can be used to omit certain voxels from being displayed. For instance, by only changing the transfer function or the thresholding, a medical volume data set can be visually mapped to many distinct structures, e. g., the skeleton only, the muscles only or the skeleton with surrounding tissue. Obviously this structural variance has a strong impact on the light interaction between the structures of such a data set, rendering the currently known surface-based illumination techniques insufficient for interactive volume illumination. Since transfer function as well as thresholding are altered frequently it should be possible to perform these changes interactively [KKH02]. Another challenge is the fact that volume rendering requires to compute light interactions for several samples along a viewing ray and thus introduces a higher level of complexity compared to computing light interactions only once for each fragment as is necessary when rendering opaque polygonal data.

In this paper we present a method which realizes dynamic ambient occlusion as well as an approximation to color bleeding when rendering volumetric data sets. Our method is independent of the currently applied transfer function as well as the thresholding and therefore represents ambient occlusion as well as color bleeding like effects for all combinations of structures contained in a volume data set, which can be extracted interactively by changing these rendering parameters (see Figure 1). However, the presented rendering algorithm is not based on the physics of light propagation, but provides a visually convincing approximation. It can be applied to direct volume rendering (DVR) as well as isosurface shading techniques, and for the latter the isovalue can be changed interactively. Rendering time is kept low, since the proposed technique requires only little overhead compared to the solution of the standard volume rendering integral combined with the application of strictly local illumination. Besides the transfer function and the thresholding, lighting as well as the camera parameters can be changed interactively. Furthermore, in contrast to frequently used surface-based illumination models our technique does not necessarily require a gradient calculation and is therefore also applicable to homogeneous regions.

The paper is structured as follows. In the next section we will discuss related work with special consideration of volume illumination models. In order to consider light interactions between neighboring structures at interactive frame rates, we perform a two-step pre-computation, which is described in Section 3. The interactive rendering algorithm is presented in Section 4, and our results are discussed in Section 5 before concluding in Section 6.

2. Related Work

Levoy was the first who presented the commonly used model for light transport in volumetric data sets [Lev88]. His model utilizes emission and absorption as well as surface shading based on local gradients. Max has reviewed several illumination models extending the basic model for volumetric data sets [Max95]. These additionally incorporate shadowing and single scattering as well as multiple scattering effects. Max states that light interactions between neighboring structures are important for volume rendering. Due to performance restrictions these effects cannot be found in interactive volume rendering applications yet, but some work has been done to support illumination based on the physics of light propagation offline. Kajiya and von Herzen have proposed a ray tracing approach for volumetric data sets which allows also to incorporate a solution for their derived scattering equation [KH84]. Sobierajski and Kaufman present a ray tracing algorithm which can render scenes containing volumetric as well as geometric data and incorporates light interactions as shadows and reflections [SK94]. To target interactivity, Parker et al. have proposed a brute-force ray tracing system for isosurface shading and MIP rendering of volumetric data, which runs on a multiprocessor system [PPL*99]. However, like with the previously proposed methods [KH84, SK94], no color bleeding can be achieved, and even today the frame rates would not be interactive on conventional machines. Rushmeier and Torrance have developed the zonal method, an extension to the radiosity technique which simulates scattering as well as diffuse interreflections when surfaces and participating media are present [RT87]. Radiosity methods solely for volumetric data incorporating diffuse interreflections have been proposed by Avila et al. [AHH*94] and Sillion et al. [Sil94]. Both methods are based on a physical model of light transport and thus generate visually convincing results. However, they do not support interactive modification of important rendering parameters, e. g., the

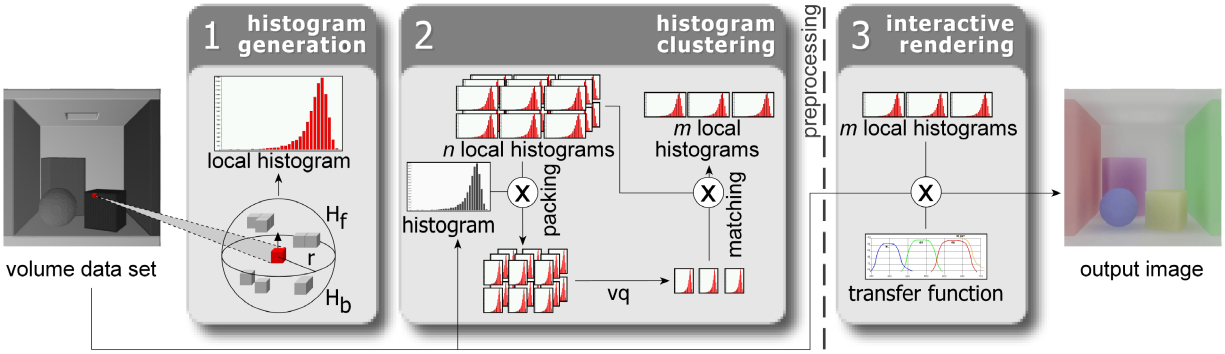


Figure 2: ***Workflow****: In the first preprocessing stage a local histogram is generated for each of the n voxels in order to capture the distribution of intensities in its environment (1). Then the n local histograms are sorted into m clusters ($m < n$) through a vector quantization (vq). To accelerate the vector quantization, it operates on packed histograms. The packing is based on the histogram of the volumetric data set. After the clustering is finished the packed local histograms are replaced by their unpacked counterparts during the matching, before computing new cluster representatives (2). During rendering the local histograms representing the clusters are modulated with the transfer function to determine an environmental color for each voxel (3).*

thresholding or the transfer function. More recently, interactive volume illumination models have been proposed which simulate global effects. Similar to the technique presented within this paper, these approaches are based on approximations of light interactions. Kniss et al. have presented a model which approximates scattering and chromatic attenuation [KPH*03]. In order to support scattering they consider the neighborhood of each voxel during rendering, leading to a high performance impact. Vicinity Shading [Ste03] simulates illumination of isosurfaces by taking into account neighboring voxels. In a pre-computation the vicinity of each voxel is analyzed and the resulting value, which represents the occlusion of the voxel, is stored in a shading texture which can be accessed during rendering. However, in contrast to our approach Vicinity Shading requires a new preprocessing when changing the rendering parameters, and it does not support color bleeding. Desgranges and Engel describe a less expensive approximation of ambient light than Vicinity Shading [DE07]. They combine ambient occlusion volumes from different filtered volumes into a composite occlusion volume. While pre-processing time is greatly reduced the ambient occlusion volume still must be recomputed whenever the transfer function is changed. Recently Hernell et al. have proposed a method for computing ambient and emissive tissue illumination efficiently [HLY07]. In contrast to our approach, they exploit ray casting to determine the ambient illumination, and they have to deal with LoD volumes to speed up rendering. Wyman et al. have presented a technique to pre-compute or lazily compute global illumination for interactive rendering of isosurfaces extracted from volumetric data sets [WPHS06]. They support the simulation of direct lighting, shadows and interreflections by storing pre-computed global illumination in an additional volume to allow viewpoint, lighting and isovalue changes. Beason et al. present a method which additionally can represent translucency and caustics but supports static lighting only [BGB*06]. For that purpose they extract different isosurfaces from a volumetric data set, illuminate them with a path tracer and store the results in a new volume data set. During rendering they can interactively change the isovalue and access the pre-computed illumination. All these surface illumination models are only applicable to isosurfaces representing a single intensity within the data set, but do not allow to consider multiple surfaces corresponding to different intensities. Hence it is not possible to represent the entire volume data set, whereas varying intensities are one of the major advantages over polygonal models, e. g., when representing different types of tissue.

3. Pre-Computation and Storage of Light Interaction

In this section we describe our two-step pre-computation necessary to capture light interactions between neighboring structures within a volumetric data set. In the pre-computation we ensure that we analyze and store the environment of each voxel x in such a way that we are able to compute an environmental color E_{env} — approximating the influence of the voxels neighborhood — during rendering interactively. In order to support interactive modification of the transfer function and the thresholding, the computation is performed independently of these rendering parameters. The consecutive steps needed are shown in Figure 2 and are further explained in the following subsections.

Throughout the rest of this paper we assume that a volumetric data set is representing a scalar intensity function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, which assigns each point x a scalar value $f(x)$. In our case the volumetric data set defines f by its values on a regular discretized grid. For points not lying on the grid, the

Preliminary Version.
The definite version will be available at diglib.eg.org
and www.blackwell-synergy.com.

value of f can be computed by an appropriate interpolation. Transfer functions are used to map intensities to emissive colors as well as opacities, and the gradient for the current voxel x can be computed by considering the transfer function and is denoted by $\bigtriangledown\tau(f(x))$.

3.1. Local Histogram Generation

To approximate the environmental color E_{env} for a given voxel x, we exploit its local histogram $LH(x)$. Local histograms have also been used in other areas of volume rendering [RBS05, Lun06]. For our approach local histograms are adequate, because indirect illumination can be calculated properly for a given point by considering close objects only [DS06]. All voxels $\tilde{x}$ lying in a sphere $S_r(x)$ with radius r centered around x contribute to the local histogram $LH(x)$, weighted based on their distance to x (see step 1 in Figure 2). Thus assuming that $f(x) \in [0,2^b]$, with $b \in \{8,12,16\}$ being the bit depth of the data set, LH assigns to each x an n-tuple, with $n = 2^b$:

$$LH(x) = (LH_0(x), \ldots, LH_{n-1}(x)), \text{ with} \tag{1}$$

$$LH_k(x) = \sum_{\substack{\tilde{x} \in S_r(x) \\ \tilde{x} \neq x}} f_{dist}\left(\frac{|x-\tilde{x}|}{d_{min}}\right) \cdot g(f(\tilde{x}),k). \tag{2}$$

d_{min} denotes the minimal distance between any two different voxels in the data set. $f_{dist} = \frac{1}{d^2}$ is used to achieve a distance based weighting and takes into account that energy falls off as the inverse square of the distance. $g(i,k)$ is used to group the intensity values appropriately:

$$g(i,k) = \begin{cases} 1\,, & \text{if } i = k \\ 0\,, & \text{otherwise.} \end{cases} \tag{3}$$

$LH_k(x)$ represents the influence voxels of intensity k in the neighborhood of x have on x. In contrast to other techniques we can not consider the attenuation of light which results from voxels lying between the current voxels and its neighbors, because we discard the spatial locations. Since we are only interested in the relative distribution of $LH(x)$, we normalize the values in each $LH(x)$ with respect to the number of voxels lying in the sphere $S_r(x)$ with radius r centered around x.

To capture the neighborhood of an object in scenes consisting of polygons, often ray casting is exploited which involves sampling that may influence the image quality. Since volume data sets are already a discretized representation, ray casting and thus possible sampling artifacts should be avoided. We use a simple method which captures the vicinity of voxel x by iterating over all voxels $\tilde{x}$ in its neighborhood and adding their contribution to $LH(x)$ as defined by equation (2). By definition $LH(x)$ does not contain any spatial information except the distance based weighting f_{dist} inherently capturing the degree of influence of the voxels in the neighborhood. In order to capture also directional information, we subdivide $S_r(x)$ based on the gradient at x into two hemispheres. Instead of one local histogram for $S_r(x)$, we compute two local histograms, one for the forward facing hemispherical region $H_f(x)$ and one for the backward facing hemispherical region $H_b(x)$ (see Figure 2). Rendering for this region-based approach is explained in Section 4.

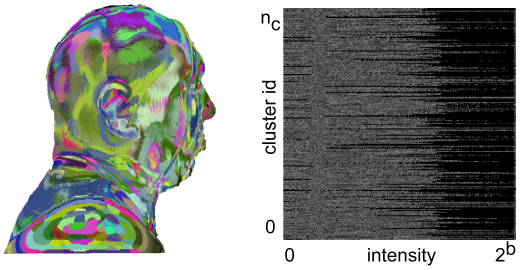


Figure 3: ***Results of the preprocessing.*** *A visualization of the volume storing the identifiers of the local histogram clusters (left), and the corresponding clusters, color coded using a black-to-white ramp function, i. e., bright colors represent a high occurrence (right).*

As already mentioned, instead of capturing the light interactions between all voxels of a data set, we consider only the vicinity defined by the radius r. Obviously r is data set dependent, but can generally be chosen rather small in comparison to the number of voxels n. This reduces the complexity from $O(n^2)$ operations to $O(r^3 \cdot n)$. The influence of r on the preprocessing performance is discussed in Section 5.

3.2. Local Histogram Clustering

With the method described in the preceding subsection we have generated two normalized local histograms for each voxel x. To make this large amount of information efficiently accessible during rendering we perform a clustering in order to reduce the number of local histograms we have to consider during rendering (see step 2 in Figure 2). The result of the clustering is a smaller set of local histograms, which we denote as cb (= code book), and a volumetric data set, which contains identifiers associating each voxel x with two elements of cb (see Figure 3). The elements of cb are not necessarily a subset of the initially computed set of local histograms, but are averaged over all local histograms contained in the cluster they represent.

For measuring the similarity of histograms, bin-to-bin and cross-bin techniques can be used. To achieve good image quality we have to keep the different intensity values in each

bin of the local histograms separated during the comparison. Therefore we have chosen a bin-to-bin clustering method, for which we interpret each normalized $LH(x)$ as a vector having dimensionality 2^b. Using the L_2 distance metric in the resulting 2^b-dimensional vector space satisfies the requirements of a bin-to-bin comparison. We use a vector quantization to cluster the 2^b dimensional vectors. In particular we have implemented the well known LBG algorithm [LBG80] with some slight modifications, which are explained below. The vector quantization training phase is performed using a subset TS of the original elements. The cardinality of TS and its influence on the achieved image quality are briefly discussed in Section 5. Instead of choosing the elements of TS randomly they can be chosen with consideration of their characteristics and the desired output image. Thus we could achieve improved results for simulating color bleeding on surface-like structures when considering only voxels x with $|\bigtriangledown f(x)| > \varepsilon$.

The runtime complexity of the vector quantization depends on both the number of desired clusters as well as the dimensionality of the underlying vector space. Even when using the L_1 metric for comparing cluster elements in the training phase of the vector quantization, it would be still computationally expensive. One major bottleneck is that the comparison of each element with every cluster during the training phase has a time complexity of $O(k^3)$, k being the number of desired clusters. However, reducing k has a strong impact on the image quality since compression artifacts arise in the final image, as we will show in Section 5. Therefore instead of reducing the number of clusters we speed up the comparison of two local histograms by reducing the dimensionality of the underlying vector space. This can be achieved by using *histogram packing*. To find a sufficient packing scheme we take into account the normalized histogram H of the initial volume data set. Based on this packing scheme we iteratively split the bins of all computed local histograms and distribute their content to the neighboring bins. We determine the packing scheme by iteratively obtaining a new histogram H_i by splitting the smallest bin of a histogram H_{i-1} and distributing it equally among the two neighboring bins. We apply as many splitting iterations as necessary to obtain the desired number of bins equal to the dimensionality of the lower dimensional vector space. The bin numbers used are also briefly discussed in Section 5. Once the packing scheme has been determined, it can be applied to all local histograms, and we can perform the vector quantization procedure in a vector space of lower dimensionality. For being able to derive a clustering in the higher dimensional vector space, it is important that each packed local histogram has a reference to its unpacked predecessor. Once the quantization has been performed for the lower dimensional vector space, each packed local histogram is replaced by its unpacked predecessor in order to generate high dimensional clusters containing the original histograms. After we have assigned each local histogram to a cluster, we calculate a representative for each cluster by averaging over its elements and store the resulting cluster indices for each voxel.

4. Interactive Rendering

Based on the described pre-computations we are able to integrate dynamic ambient occlusion as well as color bleeding as an approximation to diffuse interreflections into interactive volume rendering systems. In comparison to Blinn-Phong shading commonly used in interactive volume rendering, we need only two additional texture fetches for each sample to determine the color in its neighborhood specified by the radius r: One in a 3D texture to get the cluster id for the current voxel, and one in a 1D lookup texture to get the environmental color E_{env} associated with the determined cluster. This lookup texture has to be updated whenever the transfer function is changed in order to visualize different structures inherently contained in a data set. The update process of the 1D lookup texture is described in the next Subsection 4.1. In the following two Subsections 4.2 and 4.3 we will describe the application to isosurface shading as well as DVR, which can be performed efficiently in a fragment shader. Furthermore we propose a simple extension which can capture volumetric glow in Subsection 4.4.

For simplicity we assume in all subsections that the thresholding has been integrated into the transfer function in such a way that all intensity values lying outside the threshold interval are mapped to zero opacity.

4.1. Environmental Color Update

To update the environmental color E_{env} at a voxel x used during rendering, we have to take into account the currently used transfer function as well as the local histograms representing the pre-computed clusters (see step 3 in Figure 2). Since this update will happen frequently, i. e., whenever the transfer function is changed, we have to ensure that it can be performed efficiently. We determine the environmental color at x for the hemisphere determined by the gradient $\bigtriangledown\tau(f(x))$ from the corresponding histogramm $LH(x)$ as follows:

$$E_{env}(x, \bigtriangledown\tau(f(x))) = \frac{1}{\frac{2}{3}\pi r^3} \sum_{0 \le j < 2^b} \tau_\alpha(j) \cdot \tau_{rgb}(j) \cdot LH_j(x). \tag{4}$$

$\tau_{rgb}(j)$ evaluates to the emissive color of voxels having the intensity j, and $\tau_\alpha(j)$ evaluates to their opacity. We consider $\tau_\alpha(j)$ to ensure that opaque voxels have a stronger contribution to $E_{env}(x)$ than more translucent ones.

In cases were no color bleeding is desired, we simply take into account the occlusion of the neighborhood, as it is

known from conventional ambient occlusion techniques. For these cases we compute an occlusion factor O_{env} as follows:

$$O_{env}(x, \bigtriangledown\tau(f(x))) = \frac{1}{\frac{2}{3}\pi r^3} \sum_{0 \leq j < 2^b} \tau_\alpha(j) \cdot LH_j(x). \quad (5)$$

Again the $\tau_\alpha(j)$ coefficients ensure that a voxel's contribution is proportional to its transparency specified by the transfer function. E_{env} as well as O_{env} are based on the gradient $\bigtriangledown\tau(f(x))$ of the current voxel x. For voxels belonging to surface-like structures it poses no problem to derive $\bigtriangledown\tau(f(x))$, and the aforementioned definitions of E_{env} and O_{env} can be used. However, a major benefit of volumetric data sets in comparison to polygonal models is the capability to capture homogeneous regions. Since for these regions no sufficient gradient can be computed, alternative strategies have to be considered. We assume that the major light phenomenon within these homogeneous regions is based on scattering. In order to simulate scattering, we consider not only the forward facing hemisphere $H_f(x)$ surrounding the current voxel, but also the backward facing hemisphere $H_b(x)$. Similar to the approach described in [KPH*03], we use the Henyey-Greenstein phase function to achieve the desired scattering effect in homogeneous regions. This function can be written as $P(\omega, \omega')$ and defines the amount of energy coming from direction ω' that is scattered towards ω. By setting ω' to be the light vector and ω to be the view vector, we can use the result of P to modulate the environmental color E_{env} for both hemispheres $H_f(x)$ and $H_b(x)$ separately, and combine the results. For voxels lying on the border of a homogeneous region, we can also apply the phase function for one hemisphere only and thus are able to achieve an effect which is visually similar to subsurface scattering (see Figure 7).

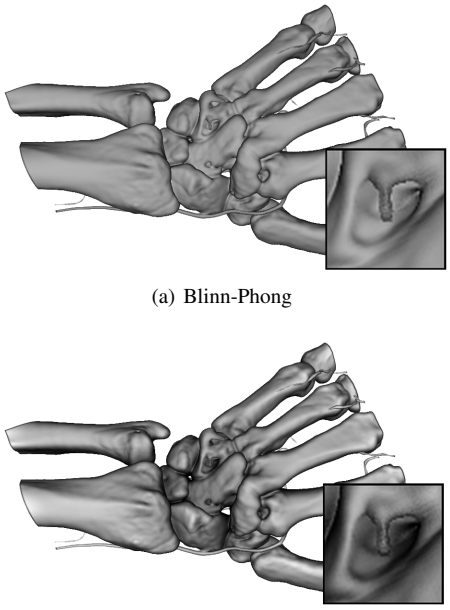
To allow interactivity, we update each histogram cluster as soon as the transfer function is changed during runtime. Even for 2048 clusters and $b = 12$ we can ensure real-time updates of the lookup texture by using equation (4).

4.2. Isosurface Shading

When using isosurface shading only voxels having an intensity equal to the isovalue are visible in the rendering. Thus only one hue is present, namely the one associated with the specified isovalue. Color bleeding is not perceivable and thus needs not be simulated. Hence only the occlusion factor O_{env} is needed for rendering. The desired effect can be achieved easily by modifying the ambient term of the Blinn-Phong model (see Figure 4):

$$I_a(x) = 1.0 - O_{env}(x, \bigtriangledown\tau(f(x))) \cdot Col_{iso}, \quad (6)$$

where Col_{iso} is the color associated with the currently set isovalue.



(a) Blinn-Phong

(b) our technique

Figure 4: *A hand data set (244 × 124 × 257 voxel) rendered using our surface shading technique (r=24, n_c=2048) in comparison to Blinn-Phong shading. Notice the shading differences in the obscured areas.*

Thus when rendering isosurfaces, similar to the work described in [WPHS06], we achieve an ambient occlusion representation, which adapts interactively to the currently set isovalue as well as isocolor. This requires only two additional texture fetches for each voxel to determine its occlusion during rendering, and we are able to change the isovalue interactively to facilitate on-the-fly surface extraction.

4.3. Direct Volume Rendering

In contrast to isosurface shading, when using DVR more than one hue contributes to the final image, since voxels with different intensities are rendered. Furthermore besides possible surfaces large homogeneous regions of participating media may be visible in the final rendering. Thus we have to be able to simulate besides ambient occlusion also color bleeding on the rendered structures, as it is demonstrated with the Cornell box data set shown in Figure 5.

To achieve this effect we determine the color for every voxel x in YUV color space, which allows us to compute hue and luminance separately. We get the hue by interpolating between the environmental color E_{env} and the color $\tau_{rgb}(x)$, which is acquired by accessing the transfer function. When assuming that a voxel which has many neighboring structures will more likely show the resulting color bleeding effects, it becomes clear that O_{env} can be chosen as the inter-

polation parameter. To get the luminance value we take the minimum of $1.0 - O_{env}$ and the Lambert term $\tau(f(x))) \cdot L$, with L being the normalized light vector. Because light adds up linearly we can integrate specular reflections by simply adding the specular intensity to the luminance.

4.4. Volumetric Glow

As mentioned above, the proposed technique can be used also for generating a volumetric glow effect. This glow effect is basically a distance function, which colors a voxel based on the voxels in its vicinity. Such a technique is potentially beneficial for highlighting structures of interest or for visualizing *safety zones* as necessary for intervention planning based on medical volume data sets. The effect can be applied interactively and is demonstrated for the Cornell box data set in Figure 5 (bottom row).

To achieve this volumetric glow effect, we introduce a new mapping function called $h(j)$, which assigns a glow intensity lying in $[1 \ldots h_{max}]$ to each intensity value j, and we modulate each summand in equation (4) by $h(j)$.

To make the glow also visible when rendering otherwise transparent voxels, we add the normalized glow intensity of the environment around x to the α value of x in cases where voxel x would be transparent otherwise.

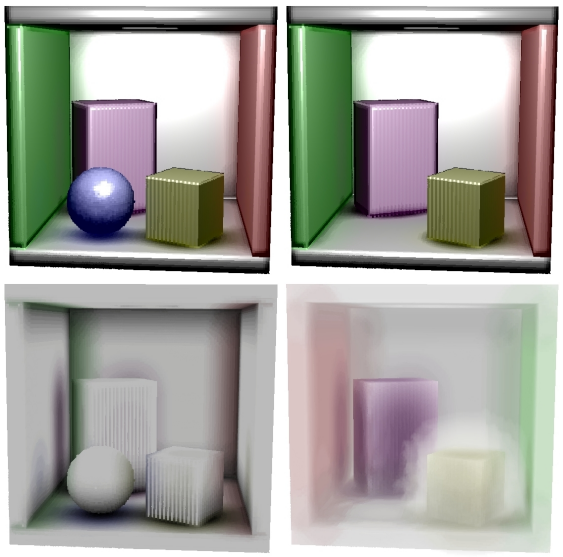


Figure 5: *All four images show the same Cornell box data set (r=32, n_c=1024). For the right column the transfer function has been changed in such a way that the blue sphere disappears. The lower row shows the images with diffuse interreflections only (left) and material parameters set to simulate highly diffuse surfaces and an additional glow effect (right). The images have been rendered interactively by changing the transfer function resp. the glow mapping. Notice the color bleeding on the objects, which disappears for the blue sphere when removing it by modifying the transfer function.*

5. Results

We have implemented the concepts described in the previous sections in C++, using OpenGL and GLSL for the image synthesis.

Preprocessing The preprocessing has been parallelized and runs on a Sun multiprocessor system equipped with eight AMD Opteron 852 processors. The needed pre-computation times are shown in Table 1. The first column shows the name of the volumetric data set, for which we have measured the performance. For each data set, we also provide its dimensions as well as the radius of the sphere analyzed to compute the environmental color E_{env}. Besides the time for computing the local histograms, we have also measured the time needed for the training phase of the vector quantization as well as the space requirements before and after the clustering has been performed, i. e., the local histogram size and the codebook size. As it can be seen, we achieved a performance gain when using four of the processors in parallel, but using more did not result in a considerable improvement. This is probably due to the shared memory bottle neck. Instead it was helpful to preprocess two different data sets simultaneously, each running with four processors. The results also indicate that the time needed for generating the local histograms is highly dependent on the sphere radius r, and unfortunately especially for large data sets a large r is necessary to achieve good results. However, in contrast to the vector quantization this preprocessing step can run in parallel on an arbitrary number of processors without much overhead and is therefore not considered the critical bottleneck. As shown in the table, exploiting spatial coherence by using the space sweep paradigm allowed for a high performance gain of about a factor of 10, as it can be seen for the head data set having a resolution of $256 \times 256 \times 147$, while getting identical results. Instead the main bottleneck is the training phase of the vector quantization. It can be seen that this step consumes a lot of preprocessing time. The vector quantization consists of several interdependent subtasks, hence not much performance is gained on a multiprocessor system. However, we were able to enhance speed of the training phase by sorting more than one vector in each iteration, without noticing any impact on the image quality.

Additionally, the time required for both preprocessing steps as well as the memory requirements can be reduced by performing the pre-computations for every ith voxel only, since the vector quantization can be accelerated drastically by using a smaller training set. Currently the memory requirements depend on the number n_c of used clusters as well as the size of the volume data set.

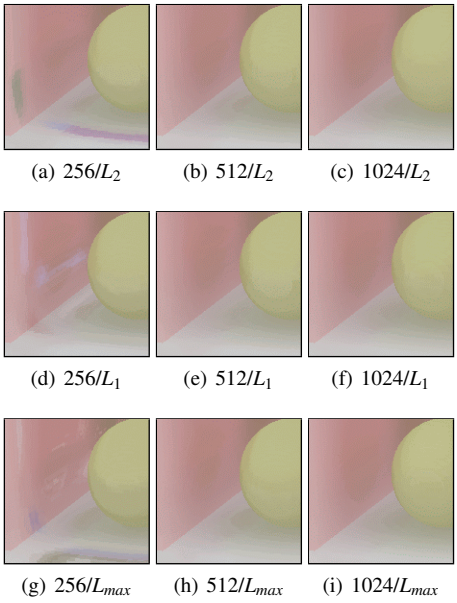
Rendering We have integrated the rendering into a GPU-

based volume ray caster. In comparison to standard Blinn-Phong shading we perform only two additional texture fetches for each sample. The first texture fetch is performed on a 3D texture and determines the cluster to which the current voxel belongs (see Figure 3 (left)). The second texture fetch is performed on a 1D texture in order to look up the environmental color E_{env} we have computed for the cluster the voxel is associated with. In cases where both hemispheres surrounding a voxel contribute to its color, e. g., in homogeneous regions, an additional texture lookup is required only in the 1D texture, since two cluster identifiers can be fetched from a 32-bit volume simultaneously.

To show the influence of different preprocessing parameters, we have applied our method to a volumetric representation of a Cornell box (see Figure 6). The images have been rendered with a different number of clusters n_c as well as using different metrics during the vector quantization. Disturbing compression artifacts are clearly visible for $n_c = 256$. Using $n_c = 512$ is more suitable for this data set, although the renderings with $n_c = 1024$ look slightly smoother. With this data set we have not observed any visual differences between $n_c = 1024$ and $n_c = 2048$. Certainly this is not true for arbitrary data sets, but using $n_c = 2048$ we have experienced good results for real-world data sets.

Figure 4 shows our surface-based technique applied to a CT scan of a human hand in comparison with Blinn-Phong shading. The results are similar to standard ambient occlusion techniques as known for polygonal data sets, although we can additionally vary the visualized structures by changing the transfer function interactively. As it can be seen, shading differences in the obscured areas become clearly visible. Especially cavities of the middle hand (see overlay) and the surface structure of the forearm bone become better perceivable, which potentially leads to an improved spatial comprehension.

We also have applied our techniques to the Visible Human head data set. The data set shown in Figure 1 has been preprocessed with a spherical region having a radius of $r = 20$ and $n_c = 2048$ clusters. The two renderings differ only in the transfer function. It can be seen that the skull structures still show specular highlights on the unoccluded surfaces, while light dimming becomes visible in regions with a higher degree of occlusion, i. e., between the vertebrae and in the cavities of the jaw. The image on the right for which the transfer function has been altered interactively to visualize additional tissue structures also shows the influence of the skullcap on adjacent parts of the brain. Figure 7 shows our technique applied to the Visible Human head data set in comparison to Blinn-Phong shading. In order to incorporate the subsurface scattering effects of the human skin, we have not only considered occlusion, which is given by the forward facing hemisphere $H_f(x)$, but have also considered the backward facing hemisphere $H_b(x)$, as described in the previous section. Additionally ambient occlusion effects become visible in the inner parts of the auricle.



(a) 256/L_2 (b) 512/L_2 (c) 1024/L_2

(d) 256/L_1 (e) 512/L_1 (f) 1024/L_1

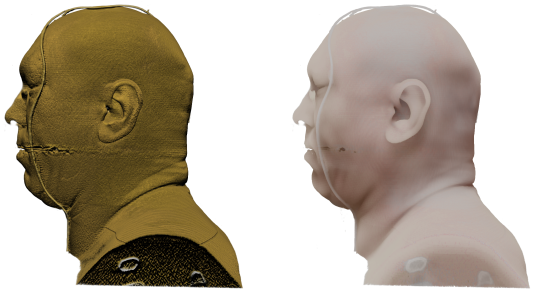
(g) 256/L_{max} (h) 512/L_{max} (i) 1024/L_{max}

Figure 6: *Magnifications of the Cornell box shown in Figure 5 with different numbers of clusters n_c (from left to right: 256, 512, and 1024) combined with different metrics used for the histogram comparison during the vector quantization (from top to bottom: L_2, L_1, and L_{max}). Only the current voxel's color darkened according to the local occlusion O_{env} is shown.*

Due to the nature of the proposed algorithm, only light interactions between neighboring structures can be captured. Although this is appropriate to give the user the impression of color bleeding, for some application cases extended light interactions may be required. For these cases, our technique can be easily combined with existing approaches as for example interactive shadowing techniques.

6. Conclusions and Future Work

In this paper we have proposed a method for rendering volumetric data sets, which is capable to capture dynamic ambient occlusion as well as color bleeding at interactive frame rates. After performing a preprocessing, volumetric data sets can be rendered at interactive frame rates still allowing the user to change all relevant rendering parameters, i. e., the viewpoint, lighting parameters, thresholding as well as the transfer function. To our knowledge this method is the first supporting the simulation of dynamic ambient occlusion as well as color bleeding at interactive frame rates when rendering volumetric data sets. Although our technique is restricted to capture light interactions between ad-



(a) Blinn-Phong (b) our technique

Figure 7: *Our interactive volume rendering method (r=20, n_c=2048) applied to the Visible Human head data set (192 × 192 × 110 voxel). Both hemispheres, in direction of the gradient and the opposite direction, are considered during rendering, and thus in contrast to Blinn-Phong subsurface scattering effects as well as ambient occlusion in the inner parts of the auricle.*

jacent structures only, it can be combined easily with existing approaches, e. g. shadowing techniques. We have explained the necessary preprocessing and have shown how our method can be applied to isosurface shading as well as DVR. Furthermore we have extended our method to support a volumetric glow effect and discussed our results. Since the proposed approach is easy to implement, we hope it finds its way into existing volume rendering applications in order to support spatial comprehension.

Nevertheless, it should be emphasized that the presented approach also has some drawbacks. Besides the time-consuming pre-processing our method is just an approximation to ambient occlusion since we do not consider the location of the voxels contributing to the local histograms. Furthermore, since we analyze just local histograms, we cannot capture true global illumination effects, i. e. distant objects do not effect the illumination.

In the future additional research can be done targeting several extensions to the proposed method. The most obvious research direction would be to reduce the time required for performing the pre-computation. Since especially the vector quantization is time-consuming we will consider alternative strategies. At the moment we are investigating in how far the vector quantization can be accelerated by exploiting a GPU-based implementation. Furthermore, the radius r of the spheres used during generation of the local histograms is data set dependent. It should be investigated what characteristics of a data set, e. g., size and contained structures, should be taken into account when choosing r.

Additionally it should be evaluated what limitations the proposed approach has. Although we did not encounter any visual artifacts yet, especially when rendering isosurfaces, there might be cases in which the disregard of the spatial locations within the sphere around a voxel has an influence. Therefore it should be evaluated for which data sets and rendering parameters this might be the case.

Acknowledgements

This work was partly supported by grants from the Deutsche Forschungsgemeinschaft (DFG), SFB 656 MoBil Münster, Germany (project Z1). The presented concepts have been integrated into the Voreen volume rendering engine (www.voreen.org). The Visible Human is an anatomical data set provided by the National Library of Medicine. Additionally, we would like to thank the reviewers for their helpful comments.

References

[AHH*94] AVILA R., HE T., HONG L., KAUFMAN A., PFISTER H., SILVA C., SOBIERAJSKI L., WANG S.: VolVis: a diversified volume visualization system. In *VIS '94: Proceedings of the conference on Visualization '94* (1994), IEEE Computer Society Press, pp. 31–38.

[BGB*06] BEASON K. M., GRANT J., BANKS D. C., FUTCH B., HUSSAINI M. Y.: Pre-computed illumination for isosurfaces. In *VDA '94: Proceedings of the conference on Visualization and Data Analysis '06 (SPIE Vol. 6060)* (2006), pp. 1–11.

[CHH03] CARR N. A., HALL J. D., HART J. C.: GPU algorithms for radiosity and subsurface scattering. In *HWWS '03: Proceedings of the conference on Graphics Hardware '03* (2003), Eurographics Association, pp. 51–59.

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

[DS06] DACHSBACHER C., STAMMINGER M.: Splatting indirect illumination. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 93–100.

[HLY07] HERNELL F., LJUNG P., YNNERMAN A.: Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *Eurographics/IEEE-VGTC Symposium on Volume Graphics* (2007), IEEE.

[KH84] KAJIYA J. T., HERZEN B. P. V.: Ray tracing volume densities. In *SIGGRAPH '84: ACM SIGGRAPH 1984 Papers* (1984), ACM Press, pp. 165–174.

[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.

Table 1: *Preprocessing times for selected data sets. While the brute-force implementation requires a lot of pre-computation time, the improved implementation exploiting spatial coherence and a faster vector quantization training achieves reasonable times. The calculations have been performed on an 8 processor system, equipped with AMD Opteron 852 processors running at 2600 MHz where each process uses 4 CPUs. Different durations for similar tasks may be caused by the non-exclusive access to the multiprocessor system.*

data set	size (voxel^3)	sphere radius	hist. gen. (min.)	training parameters (codewords/packed dim)	training (min.)	local histo–gram size	codebook size
Cornell box	$128 \times 128 \times 128$	32	236.03	256/16	2.10	2048 MB	0.01 MB
				512/16	7.08		
				1024/16	21.90		
head	$192 \times 192 \times 110$	12	16.63	2048/64	528.58	3960 MB	0.5 MB
		16	38.71		484.31		
		24	132.40		510.01		
	$256 \times 256 \times 147$	16	101.60	2048/64	704.21	9408 MB	0.5 MB
	$512 \times 512 \times 294$	32	3025.38		–[a]	75264 MB	0.5 MB
hand	$244 \times 124 \times 257$	24	514.31[b]	2048/64	633.80	7593 MB	0.5 MB
feet[b]	$128 \times 64 \times 128$	12	15.98	2048/64	320.81	1024 MB	0.5 MB
data sets computed using pre-processing with performance improved implementation (using space sweeping)							
head	$256 \times 256 \times 147$	16	10.26	2048/64	61.60	9408 MB	0.5 MB

[a] not calculated. [b] calculated on a system with 4 × Xeon 2.8.

[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.

[LBG80] LINDE Y., BUZO A., GRAY R.: An algorithm for vector quantizer design. *IEEE Communications 28*, 1 (1980), 84–95.

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

[Lun06] LUNDSTROM C.; LJUNG P. Y. A.: Local histograms for design of transfer functions in direct volume rendering. *Transactions on Visualization and Computer Graphics 12*, 6 (Nov.-Dec. 2006), 1570–1579.

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

[PPL*99] PARKER S., PARKER M., LIVNAT Y., SLOAN P.-P., HANSEN C., SHIRLEY P.: Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics 5*, 3 (1999), 238–250.

[RBS05] RÖTTGER S., BAUER M., STAMMINGER M.: Spatialized transfer functions. In *EuroVis* (2005), pp. 271–278.

[RT87] RUSHMEIER H. E., TORRANCE K. E.: The zonal method for calculating light intensities in the presence of a participating medium. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), ACM Press, pp. 293–302.

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (2003), ACM Press, pp. 382–391.

[Sil94] SILLION F.: Clustering and Volume Scattering for Hierarchical Radiosity Calculations. In *EGRW '94: Proceedings of the Eurographics workshop on Rendering* (Darmstadt, Germany, 1994), Eurographics Association, pp. 105–117.

[SK94] SOBIERAJSKI L. M., KAUFMAN A. E.: Volumetric ray tracing. In *VVS '94: Proceedings of the 1994 symposium on Volume Visualization '94* (1994), ACM Press, pp. 11–18.

[SLS05] SLOAN P.-P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (2005), ACM Press, pp. 1216–1224.

[Ste03] STEWART A. J.: Vicinity shading for enhanced perception of volumetric data. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (2003), IEEE Computer Society, p. 47.

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