Enhancing Depth-Perception with Flexible Volumetric Halos

Stefan Bruckner and M. Eduard Gröller, *Member, IEEE*

Abstract—Volumetric data commonly has high depth complexity which makes it difficult to judge spatial relationships accurately. There are many different ways to enhance depth perception, such as shading, contours, and shadows. Artists and illustrators frequently employ halos for this purpose. In this technique, regions surrounding the edges of certain structures are darkened or brightened which makes it easier to judge occlusion. Based on this concept, we present a flexible method for enhancing and highlighting structures of interest using GPU-based direct volume rendering. Our approach uses an interactively defined halo transfer function to classify structures of interest based on data value, direction, and position. A feature-preserving spreading algorithm is applied to distribute seed values to neighboring locations, generating a controllably smooth field of halo intensities. These halo intensities are then mapped to colors and opacities using a halo profile function. Our method can be used to annotate features at interactive frame rates.

Index Terms—Volume rendering, illustrative visualization, halos.

1 Introduction

The area of illustrative visualization is concerned with developing methods to enhance the depiction of scientific data based on principles founded in traditional illustration. The illustration community has century-long experience in adapting their techniques to human perceptual needs in order to generate an effective depiction which conveys the desired message. Thus, their methods can provide us with important insights into visualization problems.

Resolving the spatial arrangement of complex three-dimensional structures in an image can be a difficult task. Particularly renderings of volume data acquired by imaging modalities such as CT or MRI often suffer from this problem. As such data frequently contain many fine, overlapping structures, the resulting images often look confusing and are difficult to interpret without additional cues. For this reason artists and illustrators have long exploited the fact that the human visual system is especially sensitive to local variations in contrast by drawing halos around objects. Near the boundary of objects that are located in front of other structures the tone is locally altered: the background or partly occluded objects are slightly darkened to create the impression of depth. Similarly, bright halos are frequently placed around objects to visually detach them from the background. In the most simple case, a halo is just a gap around the edges of an object. In photography and film halos, achieved by careful placement of lights and camera, are commonly used to accentuate objects. While this technique is motivated by natural lighting phenomena such as shadows and atmospheric effects, halos in illustrations are typically overemphasized and localized to enhance occlusion cues. Halos are used in a wide variety of different styles as illustrated in Figure 1. Manifestations of the effect range from thick opaque outlines to soft darkening of the background almost undistinguishable from realistic shadows under diffuse illumination. Frequently, halos are used as a subtle way to put emphasis on certain objects. Our goal in this work is to enable the same kind of flexibility for computer-generated halos in direct volume rendering. As it is often dependent on the context of the visualization which kind of halo provides the most effective cues, we present an approach which allows interactive adjustment of their appearance.

The paper is structured as follows: Related work is discussed in Section 2. We present our approach for generating volumetric halos in Section 3. Section 4 details our implementation. Results are presented in Section 5. In Section 6 we discuss the implications of our method. The paper is concluded in Section 7.

• *The authors are with the Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria, E-mail: {bruckner|groeller}@cg.tuwien.ac.at.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Related Work

One way to add depth cues to volume rendered images is to use a more realistic illumination model. Yagel et al. [29] employ recursive ray-tracing which allows for effects such as specular reflection and shadows. Behrens and Ratering [2] add shadows to texture-based volume rendering. The model presented by Kniss et al. [12] captures volumetric light attenuation effects including volumetric shadows, phase functions, forward scattering, and chromatic attenuation. Max [19] gives a comprehensive overview of different optical models for volume rendering. The problem of increasing the physical realism is, however, that these models often lack control over the specific appearance of certain structures of interest. As they are based on actual physical laws, it is difficult to control individual visualization properties separately. Some approaches therefore use inconsistent illumination. Stewart [25] introduces vicinity shading, a view-independent model to enhance perception of volume data based on occlusions in the local vicinity of a sample point resulting in shadows in depressions and crevices. Lee at al. [14] present a system for automatically generating inconsistent lighting based on the object geometry. Kersten et al. [9] study the effect of different depth cues on the perception of translucent volumes.

Many approaches have been developed which deliberately use non-photorealistic techniques as a mechanism to improve the visualization of regions of interest. Levoy [15] was the first to propose modulation of opacity using the magnitude of the local gradient to enhance surfaces in volume rendering. Csébfalvi et al. [4] visualize object contours based on the magnitude of local gradients as well as on the angle between viewing direction and gradient vector using depth-shaded maximum intensity projection. Hauser et al. [6] propose two-level volume rendering for focus+context visualization of volume data by combining maximum intensity projection and direct volume rendering. Nagy et al. [20] combine line drawings and direct volume rendering techniques. Yuan and Chen [30] enhance surfaces in volume rendered images with silhouettes, ridge and valleys lines, and hatching strokes. Zhou et al. [31] propose the use of distance to emphasize and de-emphasize different regions. Viola et al. [27], inspired by cutaway views which are commonly used in technical illustrations, apply different compositing strategies to prevent an object from being occluded by less important structures. Krüger et al. [13] use interactive magic lenses based on traditional illustration techniques for focus+context visualization of iso-surfaces. Multi-dimensional transfer functions have been proposed to extend the classification space which allows better selection of features. Kniss et al. [11] use an intuitive direct-manipulation interface for multi-dimensional transfer functions. Lum and Ma [18] use lighting transfer functions to enable

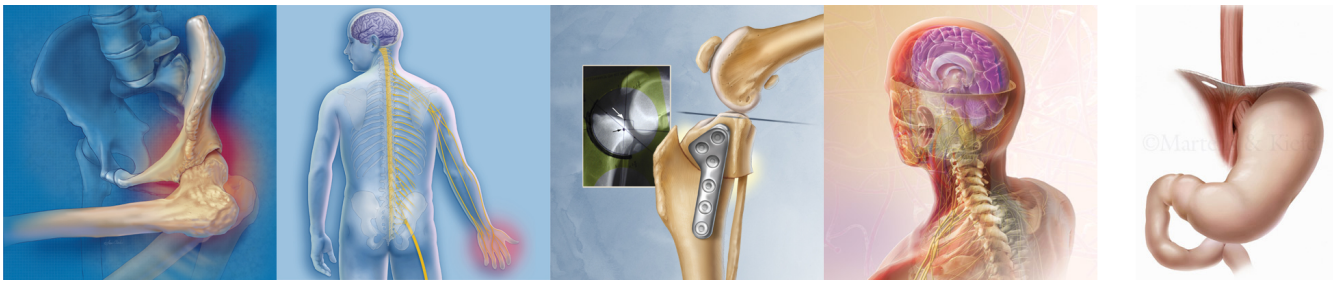


Fig. 1. Examples for the different uses of halos in medical illustration for emphasis and accentuation. The images are taken from the Medical Illustration Source Book (`http://www.medillsb.com`).

data-dependent illumination. Kindlmann et al. [10] employ curvature information to achieve illustrative effects, such as ridge and valley enhancement.

Halos and similar techniques have been used by numerous researchers to enhance depth perception. As an early example, Appel et al. [1] proposed an algorithm for generating haloed lines in 1979. Interrante and Grosch [8] employ halos to improve the visualization of 3D flow. Their approach uses line integral convolution of a texture of slightly enlarged noise spots to compute a halo volume which is then used during ray casting. Wenger et al. [28] use similar techniques for volume rendering of thin thread structures. Rheingans and Ebert [22] present feature halos for scalar volume visualization. Their approach computes an additional halo volume based on properties of the original data values. Svakhine and Ebert [26] extend this method for GPU-based volume rendering by computing the halo volume on the graphics hardware. Loviscach [16] presents a GPU-based implementation of halos for polygonal models. Ritter et al. [23] encode spatial distance in halo-like non-photorealistic shadows for the visualization of vascular structures. The approach of Luft et al. [17] is capable of enhancing surface-based images using halos by performing an unsharp masking operation on the depth buffer. Their work is a major inspiration for the approach we present in this paper, although the techniques significantly differ due to the fact that in direct volume rendering halo generation can not be performed as a post-processing step.

In this paper we contribute a new technique for generating a wide variety of halo effects using GPU-based volume rendering. Our approach classifies, generates, and maps volumetric halos on-the-fly and therefore allows flexible control over their appearance. No precomputation is required and all parameters can be modified interactively. We demonstrate that this technique is effective in enhancing depth perception in volumetric data sets without obscuring features.

3 Generating Volumetric Halos

Previous halo-generation approaches for volume rendering have frequently relied on a pre-processing step which generates a volume of halo contributions. This halo volume is then used during the rendering process to identify halo regions. The problem of this approach is that it does not allow for easy modifications of many parameters. In order to remedy this, our approach determines halo contributions during volume rendering. The algorithm operates on view-aligned slices through the volume in front-to-back order. In addition to regular sampling, classification, shading, and compositing, a halo generation pipeline is executed for every slice to process its halo contributions. The pipeline consists of three basic stages and an additional compositing step for blending the halo with the regular volume rendering. Figure 2 illustrates this process. First, regions to emit a halo are identified. We will refer to this step as halo seeding (see Section 3.1). Next, a field of halo intensity values is generated from the seeds by applying a filtering process (see Section 3.2). Finally, the halo intensities are mapped to the actual color and opacity contributions of the halo and combined with the regular volume rendering (see Section 3.3). For simplicity, the following description is only concerned with one halo. Our approach allows multiple halos, each with its own set of parameters, to be defined.

3.1 Halo Seeding

We assume a continuous scalar-valued volumetric function $f(P)$. A sample of this function at point P is denoted by f_P, the gradient vector at P is denoted by ∇f_P. For generating volumetric halos we need to classify which structures should emit halos – we call this process halo seeding. During halo seeding, a seed intensity value is generated for all samples on a view-aligned slice through f. Every point with nonzero halo seed intensity is a seed point. These seed intensity values are used in the subsequent step to derive the halo intensity values for other locations.

As halos are only drawn around the contours of objects, we need to limit our seeds to these regions. In volume rendering, contours can be characterized by the angle between the view vector v and the gradient vector ∇f_P. If these vectors are nearly orthogonal, the sample point is on a contour. Furthermore, the magnitude of the gradient vector $|\nabla f_P|$ can be used for preventing noise in nearly homogeneous regions to produce erroneous halo seeds. Using these two attributes, we can generate effective halo seeds for a given volumetric data set [22].

However, since we also want to generate localized halos which are only emitted by certain structures, we introduce a halo transfer function $h(P)$. The halo transfer function consists of several separable scalar-valued functions in the range $[0..1]$. Our approach currently supports three different components, but this could be easily extended to include, for instance, segmentation information, if available:

Value influence function $h_v(P)$. This function is based on the data value at the sample point. It is useful, for example, for generating localized halos by limiting their influence to a certain value range.

Directional influence function $h_d(P)$. This function is based on the direction of the eye-space normal, i.e., the angle between the projected gradient vector and the positive vertical axis of the image plane. It allows for directionally varying halos.

Positional influence function $h_p(P)$. This function is based on the distance of the sample point to a user-defined focus point to allow easy generation of localized halos for regions which cannot be identified solely using the data value.

The halo transfer function is then simply defined as the product of these components [11]:

$$h(P) = h_v(P)\, h_d(P)\, h_p(P)$$

The halo transfer function defines a basic seed intensity at a sample position P. This value is then combined with the gradient magnitude and the dot product between view vector and the normalized gradient vector to form the final seed intensity $s(P)$:

$$s(P) = h(P)\, |\nabla f_P|^{\alpha}\, (1 - \nabla f_P \cdot v)^{\beta}$$

where α and β are used to control the influence of the gradient magnitude and the dot product, respectively. For halos these values

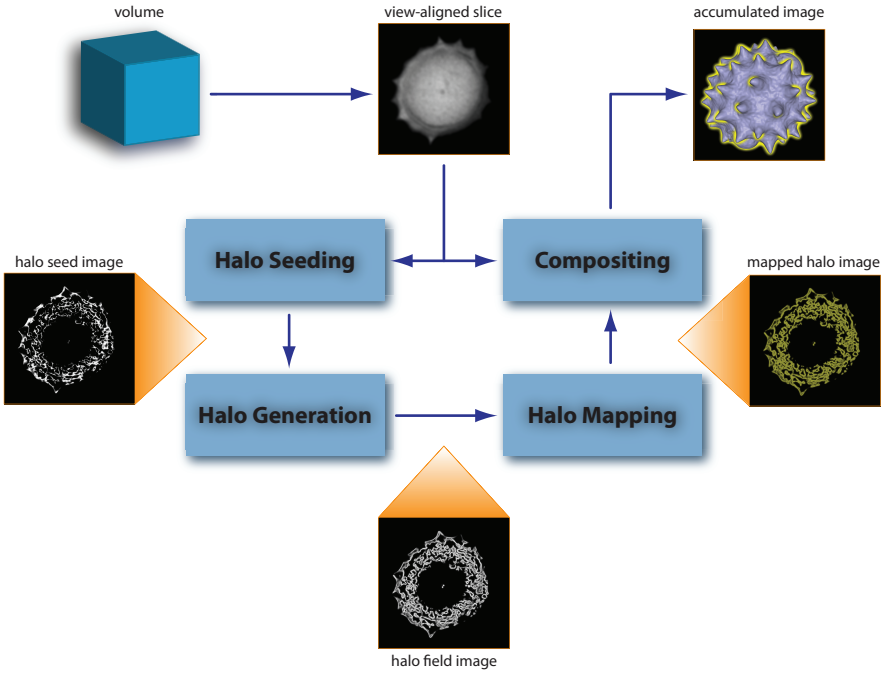


Fig. 2. Overview of the halo pipeline. The volume is processed in view-aligned slices. Halo seeding classifies halo-emitting structures, halo generation distributes the seed intensities, and halo mapping assigns colors and opacities to these seed intensities. Compositing combines the mapped halo intensities with the actual volume rendering.

are usually fixed and do not require adjustment. We use values of $\alpha = 32$ and $\beta = 0.125$ for all the images in this paper. The values of s are clamped to the range $[0..1]$. The result of applying s to all pixels of the view-aligned slice is the halo seed image S.

This definition can lead to uneven halo seeds as contours identified by dot product between normalized gradient and view vector can vary in thickness. To avoid this, we could additionally employ a modulation based on the normal curvature along the viewing direction as proposed by Kindlmann et al [10]. However, as our halo generation approach (see Section 3.2) takes special care to equalize halo contributions, we found that this is generally not necessary.

3.2 Halo Generation

Per definition halos are located outside of objects, while the halo seeds lie within other structures. Therefore, during halo generation the seed intensities are spread out to form a halo field image H. Each point within the halo field is defined by having nonzero halo intensity.

One important aspect of halo generation is the fact that halo contributions from smaller structures should not be lost during the spreading process. A naive approach would be a simple convolution of the halo seeds with a low-pass filter. This method, however, results in a reduction of halo contributions from smaller regions. The halo seeds of small structures are essentially blurred out of existence if the filter kernel is too large, although exactly those features could particularly benefit from the emphasis provided by a halo. If the kernel size is too small, on the other hand, the seed intensities are not distributed enough to generate a visible halo. This effect is illustrated in Figure 3. In Figure 3 (a) an artificial halo seed image featuring regions of multiple scale is shown. When a low-pass filter is applied to it, as shown in Figure 3 (b), the seed values are spread out, but contributions from smaller areas are lost. Other approaches such as the unsharp masking technique used by Luft et al. [17] also suffer from this problem. Thus, while acceptable from an aesthetic point of view, these methods are not suitable for our purpose.

A different approach to generating the halo field would be to perform a distance transform on the seed image. As this is computationally expensive and not well-defined on non-binary images, we realize a spreading approach which preserves the halos of small features while still generating a smooth halo field. The algorithm executes in N passes. During each pass $i \in [1..N]$ two input images are used: H_0, the initial image, and H_{i-1}, the result of the previous pass. For the first pass, the two input images are identical. For each output pixel (x,y), the algorithm first performs a convolution with a low-pass filter over the pixels of H_{i-1}. Then it combines the result of this operation with the corresponding pixel from H_0:

$$H_i(x,y) = \delta F_i(x,y) + (1 - \delta F_i(x,y))\, H_0(x,y)$$

with

$$F_i(x,y) = \sum_{u,v} w(u,v) H_{i-1}(x+ku, y+kv)$$

where w is the weight function of the filter, $k = 2^{N-i}$, and δ is a user-specified parameter in the range $[0..1]$ which controls the amount of spreading. If δ is zero the unfiltered seed image will be passed through. Increasing δ causes an increased contribution from previous passes and therefore results in a smoother, more spread-out halo while still preserving higher frequency components. Small features are preserved due to the fact that spread out values are filled up in every pass.

This algorithm is effective in distributing the halo seed values to neighboring pixels without removing high frequencies. This is visible by comparing Figure 3 (b), which uses normal low-pass filtering to Figure 3 (c), which depicts the result obtained with our approach.

However, there is still a problem as larger regions now generate a significantly larger halo. In order to remedy this, we apply the spreading algorithm to the gradient of the halo seed image instead of the original which equalizes the contributions, as illustrated in the right column of Figure 3. Figure 3 (d) depicts the gradient of the seed image. Figure 3 (e) shows that only applying a low-pass filter to the gradient image is not effective. The presented process applied to the gradient of the halo seed image, however, results in a smooth halo field with equalized contributions from structures of multiple scale, as depicted in Figure 3 (f).

If some reduction of high frequencies is desired, a median filtering could be applied to the seed image before the spreading process. However, in our experiments we found that this is not necessary in general.

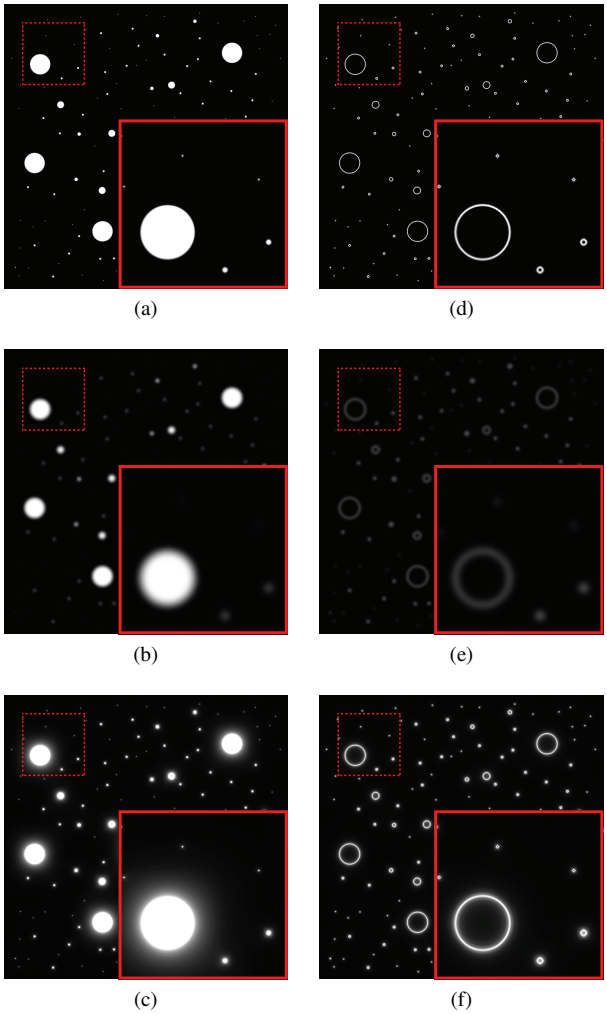


Fig. 3. Comparison of halo generation strategies on a test image. **Left column:** (a) Original halo seed image. (b) Low-pass filtered halo seed image. (c) Spreading process applied to halo seed image. **Right column:** (d) Gradient of halo seed image. (e) Low-pass filtered gradient image. (f) Spreading process applied to gradient image.

We use a filter kernel size of 3×3 with Gaussian weights. The number of iterations N determines the maximum amount of spreading that can occur – for all our purposes a value of $N = 4$ has shown to be sufficient. This algorithm is conceptually similar to the jump flooding paradigm for parallel computing [24]. Figure 4 shows results of the spreading processes for different values of δ.

3.3 Halo Mapping and Compositing

After generating the halo field image it has to be mapped to visual contributions in the image. For this purpose we employ a halo profile function: this function maps all nonzero halo intensity values to colors and opacities. Halo intensities of zero are always transparent. While the spreading parameter δ only controls the distribution of intensities in the halo field, the profile function allows further adjustment of the halo appearance.

In the simplest case, the halo profile function just maps halo intensities directly to opacities using a constant color. Other possibilities include, for instance, a halo profile with constant opacity which results in a sharp border. Figure 5 shows a few examples. In the last row of this figure, the use of directionally varying halos is also demonstrated. Finally, the mapped halo has to be combined with the volume's contribution. Based on how this combination is performed, we can distinguish between two different kinds of halos:

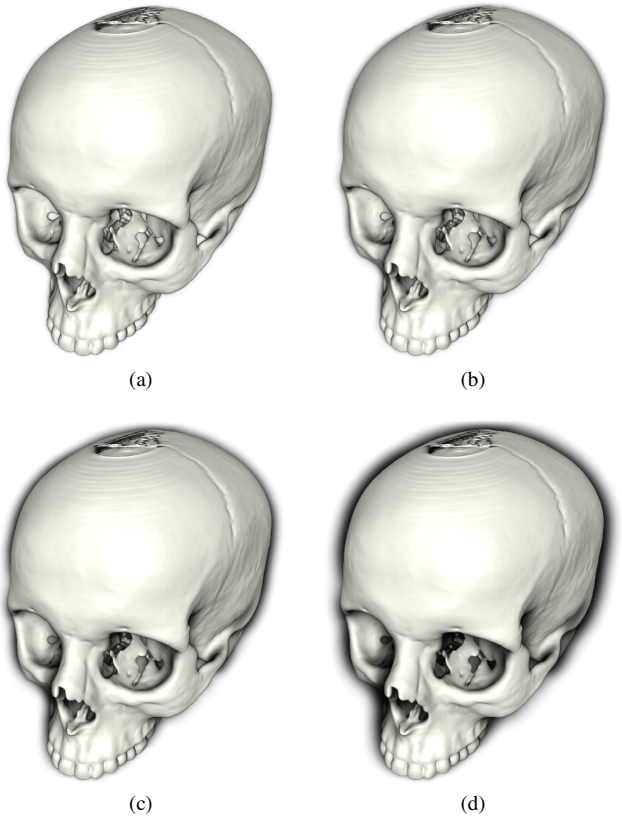


Fig. 4. Results of the halo spreading algorithm using $N = 4$ iterations with (a) $\delta = 0.70$, (b) $\delta = 0.80$, (c) $\delta = 0.90$, (d) $\delta = 0.95$.

Emissive halos. Similar to scattering of light by small particles such as fog, this type of halo causes a visible contribution by itself. From the point of view of compositing, the halo behaves as if it were part of the volume. Thus, for emissive halos the halo intensity value is first mapped using the halo profile function and then blended after the actual volume contributions using the front-to-back formulation of the over-operator. The halo therefore (partially) occludes everything located behind it including the background.

Occlusive halos. In addition to emissive halos, illustrators sometimes employ another kind of halo: the halo only contributes to the image if it occludes other structures – the halo by itself has no emissive contribution. Although similar to a shadow, this type of halo is usually drawn in the same style irrespective of the distance between the two objects and not necessarily consistent with the overall lighting conditions to strengthen the occlusion cues. This type of halo can be useful as it might be less intrusive and only highlights occlusions. For generating occlusive halos, contributions of the halo field image H need to be accumulated to be able to influence samples located behind them – this is similar to a shadow buffer [12]. For this purpose, we introduce an additional halo occlusion image O. The current halo field image H is combined with O in every pass using maximum blending. Halo mapping is then performed based on the halo occlusion image. The resulting mapped halo color is mixed with the volume sample color using the mapped halo contribution's opacity as an interpolation weight. The opacity of the volume sample remains unchanged. Thus, if no sample is occluded by the halo, it has no contribution to the image.

Both halo types are useful for different purposes. While emissive halos can be used to emphasize particular features by giving them an outline or a glow, occlusive halos provide a means for accentuating oc-

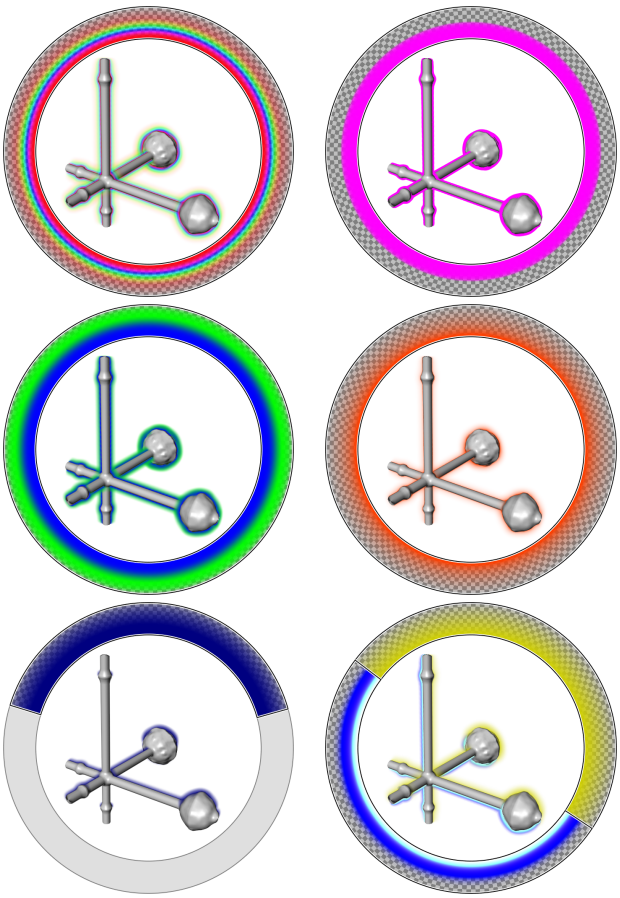


Fig. 5. Different halo profile functions applied to a simple data set. The corresponding halo profile function is shown for each image.

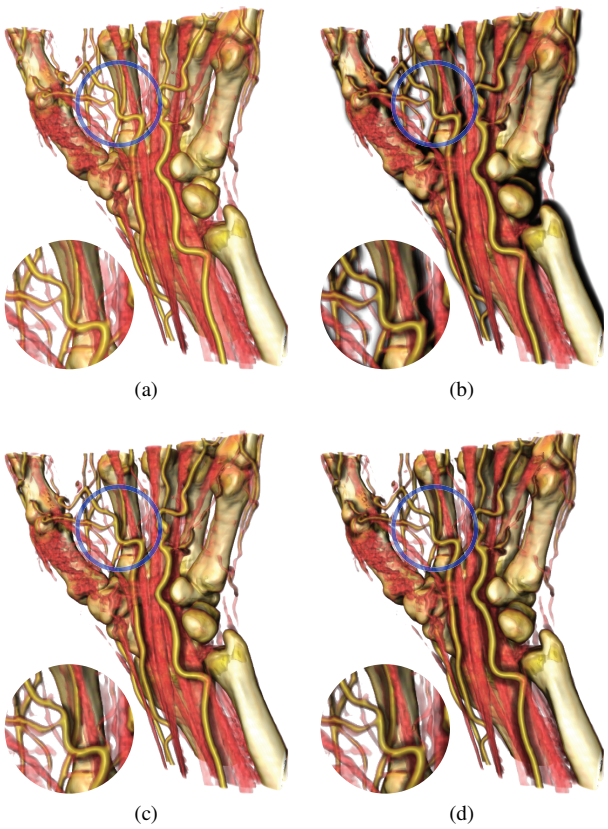


Fig. 6. Emissive and occlusive halos. (a) Volume rendering without halos. (b) A directional emissive halo is emitted by bones and vessels. (c) A directional occlusive halo is emitted by bones and vessels – it is only visible where it occludes other structures. (d) The occlusive halo emitted by bones and vessels is omnidirectional.

clusions by enhancing the contrast in areas where one object crosses another. For instance, occlusive halos are frequently used in depictions of vascular structures. Figure 6 shows a comparison of these two styles. Figure 6 (a) depicts a volume rendering without halos. In Figure 6 (b) an emissive halo is used for bones and vessels while Figure 6 (c) employs an occlusive halo. The emissive halo also generates a dark border around objects which do not occlude other structures. Figure 6 (c) and (d) also demonstrate the use of the directional component of the halo transfer function. In Figure 6 (c) a one-sided halo was generated using the directional component of the halo transfer function. Figure 6 (d) shows an omnidirectional halo for comparison. The advantage of this approach over realistic shadows lies in the fact that it can be easily adjusted without influencing the global appearance while providing emphasis in the targeted areas. Illustrators generally do not draw physically correct shadows, but they exploit the fact that the human visual system interprets this cue as an indicator for occlusion.

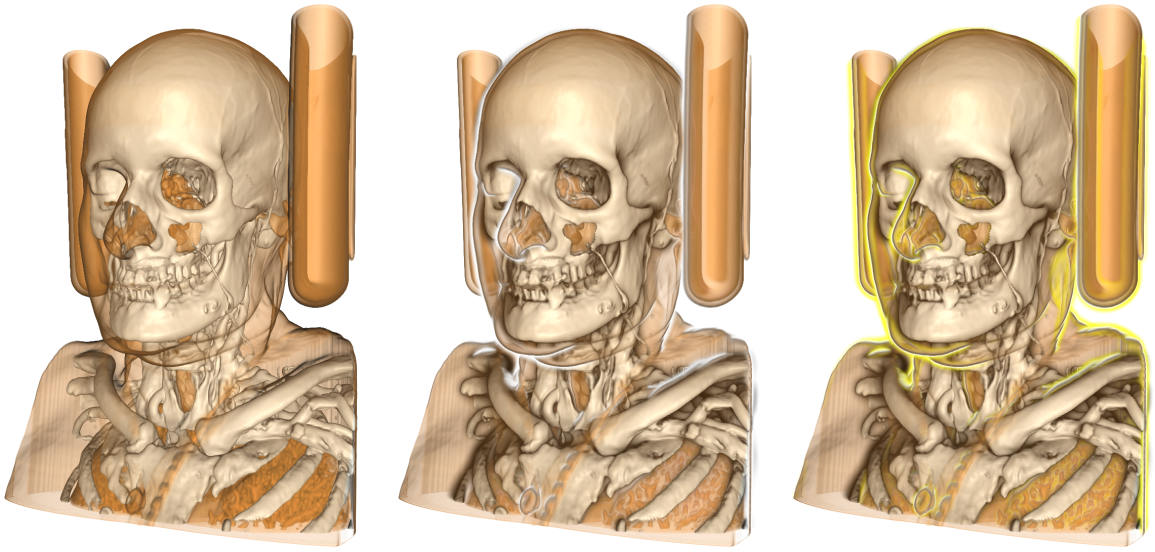
4 Implementation

Our GPU-based volume renderer with halo support was implemented in C++ using OpenGL/GLSL. As already outlined in Section 3 our approach for integrating halos with direct volume rendering is based on an interleaving of the halo generation pipeline with conventional view-aligned slicing of the volume. In this section, we discuss further details of our implementation.

Initially, six off-screen buffers are generated: I_0 and I_1 are used for compositing in a ping-pong approach. In our description, we use I_p to denote the accumulated image from the previous iteration, and I_c for the image written in the current iteration – they are swapped after each iteration. The buffer S stores the halo seed image, and H_0, H_1, and H_2 are used for halo generation. We use H to denote the buffer which contains the current halo field image, which can be either H_1 or H_2. For occlusive halos, an additional halo occlusion image O is required. We use the OpenGL ARB_framebuffer_object extension to render to and read from these buffers. The renderer slices the volume in viewing direction and has two basic phases:

Rendering. Although described separately in Section 3, as they are conceptually different stages, it is advantageous to combine halo seeding and halo mapping with regular sampling, classification, shading, and compositing in a single rendering pass. Since all these steps require the data value and gradient at the same sample location, this avoids redundant texture reads and computations and eliminates the need for an explicit mapped halo image. We take advantage of OpenGL's capabilities to write to multiple render targets in one rendering pass. The two images written to are I_c, the current accumulated image, and S, the halo seed image. First, conventional sampling, classification and shading is performed. Next, halo mapping is performed on the halo intensities read from the current halo field image H for emissive halos. For occlusive halos, the halo occlusion image O is used instead. For emissive halos, the shaded color of the volume sample is first composited with the previously accumulated color read from I_p and then written into I_c. Then the mapped halo contribution is composited. In the case of an occlusive halo the shaded sample color is mixed with the mapped halo contribution and then composited with the previously accumulated color. Finally, halo seeding is performed and the result is written to S.

Generation. This phase performs halo generation as described in Section 3.2. First, the gradient magnitude of the halo seed image is computed and written into H_0. Next, several iterations



(a) (b) (c)

Fig. 7. Improving visualization of transparent structures using halos. (a) Volume rendering without halos. (b) Depth is added to the bone by using a dark smooth halo, contours of the the transparent skin are enhanced using a white opaque halo. (c) A different halo profile function is used for the skin (fade from white to yellow).

M	frames/second	% of reference
1	4.65	15.85
2	7.72	26.31
4	10.26	34.97
8	12.59	42.91

Table 1. Performance of halo rendering measured on an AMD Athlon 64 X2 Dual 4600+ CPU equipped with an NVidia GeForce 8800 GTX GPU. The volume size was $256 \times 256 \times 256$ with a viewport size of 512×512 and an object sample distance of 0.5. The reference renderer without halos achieved 29.34 frames/second in this benchmark.

of the spreading algorithm are executed on H_0 by ping-ponging between buffers H_1 and H_2. The final halo field image is then located in buffer H_1 or H_2, depending on the number of iterations. The current halo field image H is set to this buffer. If occlusive halos are used, this image is additionally blended with the halo occlusion image O. Halo generation is the most expensive part of the rendering procedure. However, it is not necessary to perform this step for every iteration. We can use OpenGL's blending functionality to accumulate the halo seeds of several slices and then perform halo generation on this image only every M-th slice. As the seed contributions are accumulated, no features will be missed. Only if M is chosen too high, some depth accuracy is sacrificed. In practice, a value of $M = 4$ has proven to result in no visible artifacts – all figures in this paper were generated using this setting.

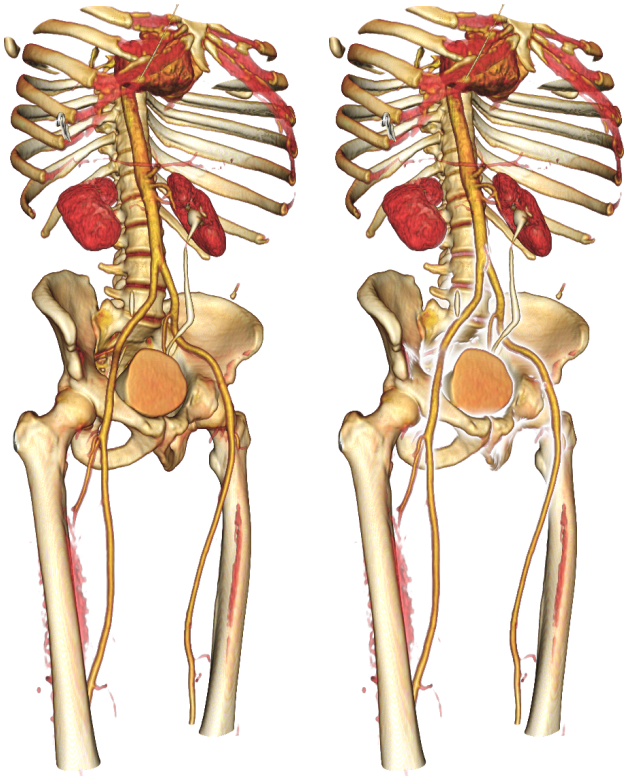
As the GPU operates on vectors rather than scalars we can support four distinct halos, each with its own set of parameters, without additional costs. Each halo is assigned a color channel and all operations, including halo generation, are simply performed on four halo channels instead of one.

5 Results

Halos provide a simple additional option for the generation of volumetric illustrations. They do not require any pre-processing and can be easily integrated into existing volume visualization tools. Similar to layer effects in image editing software such as Adobe Photoshop, they can be applied to enhance or highlight specific regions with great stylistic flexibility ranging from opaque contour-like lines to smooth object shadows. These techniques are ubiquitous in traditional illustrations and so it makes sense that volume visualization can benefit from them. In our experiments, we found that volumetric halos can be an effective and versatile tool for enhancing the visualization of volume data with little additional effort. A wide variety of data sets can benefit from halos. In this section, we present a small selection of visualization results and compare them with un-enhanced depictions.

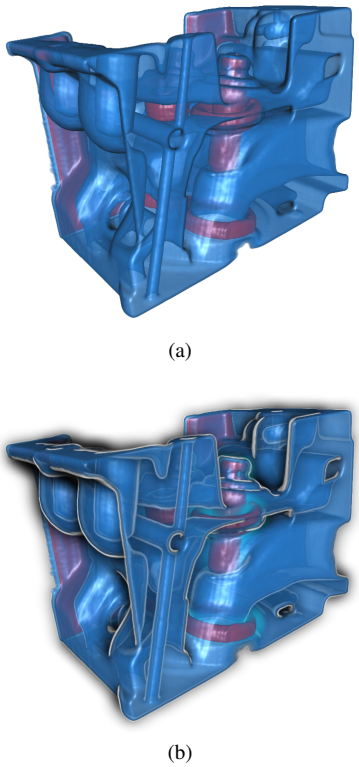
Halos are effective for the visualization of transparent objects. Illustrators frequently employ this technique in line drawings: The background object is rendered in full detail and disappears as it approaches the boundaries of the overlapping translucent structure [7]. Figure 7 demonstrates this approach: an opaque halo is used to enhance the contours of the transparent skin. Additionally, a smooth dark halo is assigned to the bone to add depth to the image. Figure 8 demonstrates the use of the positional component of the halo transfer function. A focus point is placed in the pelvic region and a bright halo is used to indicate occlusions. Through these simple means the viewer's attention is directed to the aorta. Figure 9 shows the combination of different halos for the same data value range. A smooth dark halo and a thin bright halo are assigned to the outer part of the engine block. The darkening of the white contour halo helps in indicating occlusion relationships. The inner structures additionally have a slight cyan glow. Halos may also serve as a complete replacement for normal gradient-based illumination similar to the approach described by Desgranges et al. [5]. Depending on the style of the halo, this results in an additional degree of stylization which is useful to depict contextual objects. In Figure 10 we demonstrate this effect. Figure 10 (b) and Figure 10 (c) depict different combinations of halos using shading, while Figure 10 (d) uses no additional shading. The stylized cartoon-like effect in Figure 10 (d) is achieved by a smooth background-colored halo in combination with a black contour halo.

To evaluate the performance of our implementation we compared a standard volume renderer with our algorithm. We used the UNC head test dataset (dimensions: $256 \times 256 \times 256$) and an object sample distance of 0.5. The viewport size was 512×512. No high-level optimizations such as empty-space skipping were used. Our test system was an AMD Athlon 64 X2 Dual 4600+ CPU equipped with an NVidia GeForce 8800 GTX GPU. We performed a 360 degree rotation along each axis and averaged the frame rates. The reference renderer without halos achieved 29.34 frames/second in this benchmark. The frame rates of our halo renderer for different values of M are shown in Table 1. Compared to the reference renderer, our approach achieves



(a) (b)

Fig. 8. Using halos to add occlusion cues. (a) Volume rendering without halos. (b) A white opaque halo is specified in the pelvic region using a focus point in order to accentuate the aorta.



(a) (b)

Fig. 9. Combining multiple halo effects. (a) Volume rendering without halos. (b) The semi-transparent part of the engine block uses a dark smooth halo and a white opaque halo. The inner structures have an additional cyan glow.

approximately one third of the frame rate for the typical setting of $M = 4$. Halo seeding and mapping alone, due to the increased number of texture fetches and operations per sample, result in a slowdown of about 50 percent. Although halo processing incurs an overhead, interactive frame rates are still achieved. Moreover, as our implementation is not heavily optimized, there is potential room for improvement.

6 Discussion and Future Work

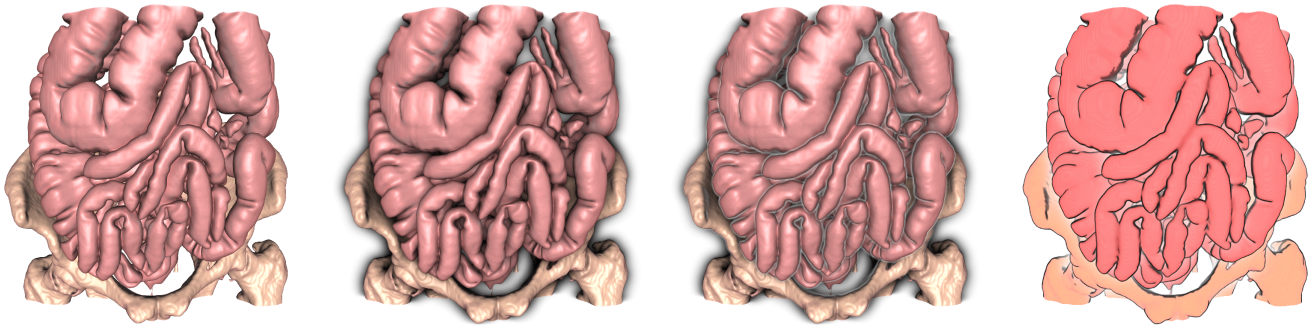
As demonstrated in this paper, halos can be used to generate easily controllable inconsistent lighting, which is not physically plausible but aesthetically pleasing. Localized shadow-like halos help to emphasize spatial relationships in an non-intrusive way as they do not change the global lighting situation. As shown by Ostrovsky et al. [21], humans are largely insensitive to such inconsistencies. Artists and illustrators therefore use inconsistent lighting to guide the viewer's attention and to enhance comprehensibility. Cavanagh [3] has suggested that our brain perceives the shape-from-shading cues only locally. This might be the reason why local cues which are inconsistent with global lighting are so effective. With volumetric halos we can generate this kind of illumination in an interactive result-oriented manner. In general, halos perform best for volumetric data which contains clearly defined features such as tomographic scans. For rather amorphous data, their benefit is limited due to the absence of distinct boundaries.

Currently, we provide a simple interactive user-interface for changing all halo parameters. The value influence function of the halo transfer function is specified using a conventional transfer function widget. The user has the option to link this function with the normal transfer function, i.e., that halos are connected to the corresponding visible structures. For the directional influence function, we provide a simple angular brushing widget which allows the specification of a range of directions on a radial layout. The positional influence function is defined by clicking on a point in the image. A viewing ray is cast from this position and records the first intersection with visible structures. The focus point is set to this position. By dragging the mouse, the inner radius and transition of the spherical focus region can be modified. The halo profile function is defined using a gradient widget. Other parameters, such as the spreading parameter δ and an additional opacity scale of the profile function can also be edited using a click-and-drag interface. While these tools are very basic, they have proven to be surprisingly effective in quickly improving the appearance of volume renderings. However, there are some enhancements that could be easily accomplished. Extending the focus point to a user-drawn polyline, for example, would be a useful extension. The resulting geometry could then be sliced in parallel to the volume and serve as an input to halo seeding. This approach would allow for even better localization of halos without the need for explicit segmentation. Other such sketch-based interaction metaphors open an interesting direction for further research.

While our interactive approach allows for quick adjustment of halo parameters, an unsolved question is which settings perform best for which types of data. A study of perceptual performance could lead to valuable insights and the resulting data could be used to derive halo templates for different scenarios.

7 Conclusion

In this paper, we presented volumetric halos as an effective and flexible method for enhancing depth perception in volume renderings. The technique was motivated by the wide-spread use of similar concepts in art and illustration. Our approach does not require pre-processing, as halo seeding, generation, and mapping are performed on-the-fly. During halo seeding, a halo transfer function is used to identify structures which emit a halo. A feature-preserving spreading procedure was discussed, which distributes halo seed intensities to neighboring sample positions. A halo profile function is used to map the resulting halo field to colors and opacities. We also introduced the notion of emissive and occlusive halos. Furthermore, we presented a GPU-based implemen-



(a) (b) (c) (d)

Fig. 10. Combining global and regional halos. (a) Volume rendering without halos. (b) A smooth shadow-like halo is used to improve depth perception. (c) An additional bright semi-transparent halo is placed around the colon. (d) Rendering without shading – a smooth white halo and a black contour halo are used to indicate the three-dimensional structure resulting in a stylized cartoon-like image.

tation of this algorithm which achieves interactive frame rates. Using several examples, we demonstrated the effectiveness of halos as a simple technique to enhance depictions of volumetric data. Based on the promising results of the presented method and due to the fact that it is easy to integrate into existing volume rendering approaches, volumetric halos can be a useful addition to the illustrative visualization toolbox.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. The work presented in this publication was carried out as part of the ex**vis**ation project (`http://www.cg.tuwien.ac.at/research/vis/exvisation`) supported by the Austrian Science Fund (FWF) grant no. P18322.

References

[1] A. Appel, F. J. Rohlf, and A. J. Stein. The haloed line effect for hidden line elimination. In *Proceedings of ACM SIGGRAPH 1979*, pages 151–157, 1979.

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

[3] P. Cavanagh. Pictorial art and vision. In R. A. Wilson and F. C. Keil, editors, *The MIT Encyclopedia of the Cognitive Sciences*, pages 644–646. MIT Press, Cambridge, MA, 1999.

[4] B. Csébfalvi, L. Mroz, H. Hauser, A. König, and M. E. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20(3):452–460, 2001.

[5] P. Desgranges, K. Engel, and G. Paladini. Gradient-free shading: A new method for realistic interactive volume rendering. In *Proceedings of Vision, Modeling, and Visualization 2005*, pages 209–216, 2005.

[6] H. Hauser, L. Mroz, G.-I. Bischi, and M. E. Gröller. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252, 2001.

[7] E. R. S. Hodges, editor. *The Guild Handbook of Scientific Illustration*. John Wiley & Sons, Hoboken, NJ, 2nd edition, 2003.

[8] V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In *Proceedings of IEEE Visualization 1997*, pages 421–424, 1997.

[9] M. Kersten, J. Stewart, N. Troje, and R. Ellis. Enhancing depth perception in translucent volumes. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1117–1124, 2006.

[10] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Visualization 2003*, pages 513–520, 2003.

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

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

[13] J. Krüger, J. Schneider, and R. Westermann. ClearView: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):941–948, 2006.

[14] C. H. Lee, X. Hao, and A. Varshney. Geometry-dependent lighting. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):197–207, 2006.

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

[16] J. Loviscach. Stylized haloed outlines on the GPU. ACM SIGGRAPH 2004 Poster, 2004.

[17] T. Luft, C. Colditz, and O. Deussen. Image enhancement by unsharp masking the depth buffer. In *Proceedings of ACM SIGGRAPH 2006*, pages 1206–1213, 2006.

[18] E. B. Lum and K.-L. Ma. Lighting transfer functions using gradient aligned sampling. In *Proceedings of IEEE Visualization 2004*, pages 289–296, 2004.

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

[20] Z. Nagy, J. Schneider, and R. Westermann. Interactive volume illustration. In *Proceedings of Vision, Modeling, and Visualization 2002*, pages 497–504, 2002.

[21] Y. Ostrovsky, P. Cavanagh, and P. Sinha. Perceiving illumination inconsistencies in scenes. *Perception*, 34(11):1301–1314, 2005.

[22] P. Rheingans and D. S. Ebert. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):253–264, 2001.

[23] F. Ritter, C. Hansen, V. Dicken, O. Konrad, B. Preim, and H.-O. Peitgen. Real-time illustration of vascular structures. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):877–884, 2006.

[24] G. Rong and T.-S. Tan. Jump flooding in GPU with applications to voronoi diagram and distance transform. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2006*, pages 109–116, 2006.

[25] A. J. Stewart. Vicinity shading for enhanced perception of volumetric data. In *Proceedings of IEEE Visualization 2003*, pages 355–362, 2003.

[26] N. A. Svakhine and D. S. Ebert. Interactive volume illustration and feature halos. In *Proceedings of the Pacific Conference on Computer Graphics and Applications 2003*, pages 347–354, 2003.

[27] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, 2005.

[28] A. Wenger, D. F. Keefe, and S. Zhang. Interactive volume rendering of thin thread structures within multivalued scientific data sets. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):664–672, 2004.

[29] R. Yagel, A. Kaufman, and Q. Zhang. Realistic volume imaging. In *Proceedings of IEEE Visualization 1991*, pages 226–231, 1991.

[30] X. Yuan and B. Chen. Illustrating surfaces in volume. In *Proceedings of Joint IEEE/EG Symposium on Visualization 2004*, pages 9–16, 2004.

[31] J. Zhou, A. Döring, and K. D. Tönnies. Distance based enhancement for focal region based volume rendering. In *Proceedings of Bildverarbeitung für die Medizin 2004*, pages 199–203, 2004.