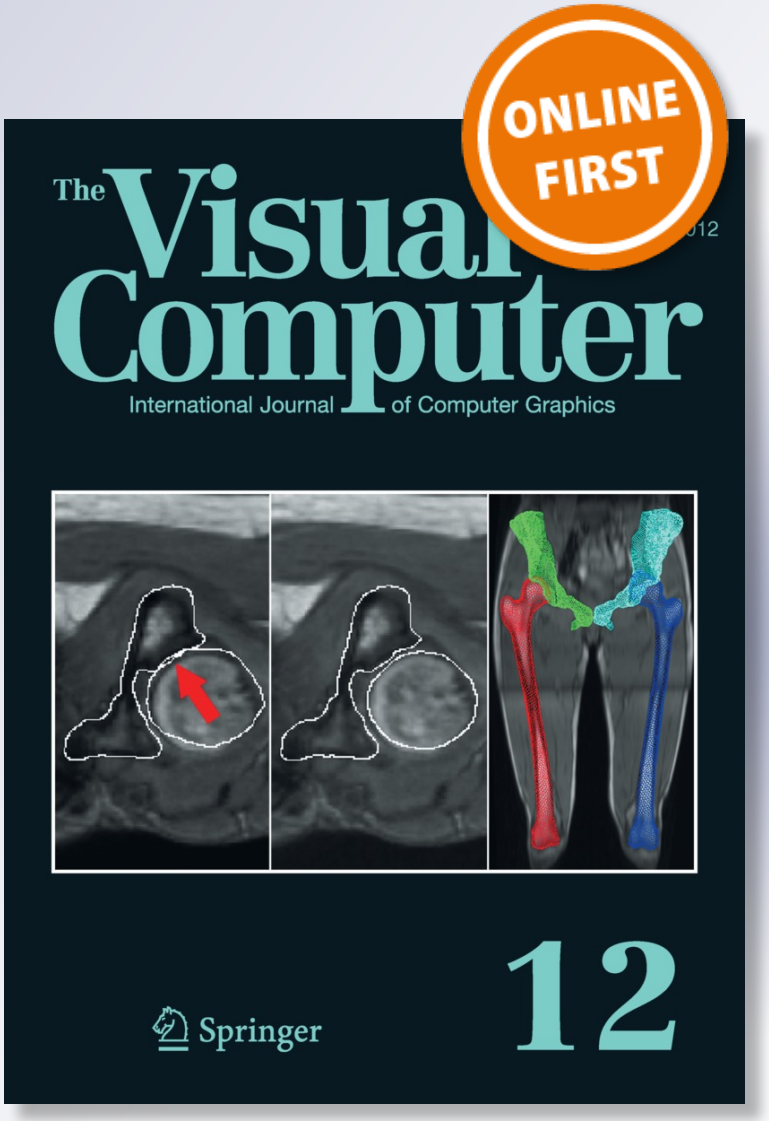
Opacity volume based halo generation and depth-dependent halos

Yubo Tao, Chao Wang, Hai Lin, Feng Dong & Gordon Clapworthy

The Visual Computer
International Journal of Computer Graphics

ISSN 0178-2789

Vis Comput
DOI 10.1007/s00371-012-0764-2



Springer

Vis Comput
DOI 10.1007/s00371-012-0764-2

ORIGINAL ARTICLE

Opacity volume based halo generation and depth-dependent halos

Yubo Tao · Chao Wang · Hai Lin · Feng Dong · Gordon Clapworthy

Abstract Halos are generally used to enhance depth perception and display spatial relationships in illustrative visualization. In this paper, we present a simple and effective method to create volumetric halo illustration. At the preprocessing stage, we generate, on graphics hardware, a view-independent halo intensity volume, which contains all of the potential halos around the boundaries of features, based on the opacity volume. During halo rendering, the halo intensity volume is used to extract halos only around the contours of structures for the current viewpoint. The performance of our approach is significantly faster than previous halo illustration methods, which perform both halo generation and rendering during direct volume rendering. We further propose depth-dependent halo effects, including depth color fading and depth width fading. These halo effects adaptively modulate the visual properties of halos to provide more perceptual cues for depth interpretation. Experimental results demonstrate the efficiency of our proposed approach and the effectiveness of depth-dependent halos.

Keywords Volume rendering · Illustrative visualization · Halos · Depth-dependent halos · Depth perception

Y. Tao (✉) · H. Lin
State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
e-mail: taoyubo@cad.zju.edu.cn

H. Lin
e-mail: lin@cad.zju.edu.cn

C. Wang
University of Wollongong, Wollongong, Australia
e-mail: w-c05@mails.tsinghua.edu.cn

F. Dong · G. Clapworthy
Centre for Computer Graphics & Visualization, University of Bedfordshire, Luton, UK

F. Dong
e-mail: Feng.Dong@beds.ac.uk

G. Clapworthy
e-mail: Gordon.Clapworthy@beds.ac.uk

1 Introduction

Illustrative volume visualization combines direct volume rendering with non-photorealistic rendering techniques to improve the visual perception of features of interest in the volume. It typically mimics and enhances traditional illustration techniques in the medical and technical domains, to assist illustrators in communicating information and knowledge more effectively. Nowadays, illustrative visualization is an active research topic in visualization [13].

A broad variety of illustrative visualization techniques have been developed to better convey volumetric structures to the user. Focus + context techniques are widely used in illustrative visualization to represent different levels of visual abstraction, such as the magic lens metaphor [23] and view-dependent interactive cutaways [21]. The shape of features can be depicted clearly by contours [14] and stippling [10], while tone shading makes orientation information easily interpretable [14]. In addition, depth perception can be effectively enhanced via feature halos [14] and volumetric halos [3]. These techniques provide illustrators with considerable artistic freedom to highlight features of interest and suppress others into part of the context.

Halos are dark or bright areas around the contours of objects that provide occlusion cues and enhance depth perception. Various halo effects have been used in traditional illustrations. In volume visualization, previous halo illustration techniques have usually performed both the halo generation

Published online: 05 December 2012

and the rendering steps during the direct volume rendering process. In fact, as this paper will demonstrate, it transpires that most of the work in halo generation can be performed at the preprocessing stage to accelerate the halo rendering.

Feature halos [14] generate a view-dependent halo volume for halo illustration and require a computationally expensive recalculation when the viewpoint is changed. Volumetric halos [3] involve a complex halo generation, i.e., halo seeding and spreading, for each projected slice in screen space during volume rendering. This results in a significant performance penalty, as halo generation requires a large number of texture fetches on graphics hardware.

In a previous paper [19], we presented a simple and effective method for the rapid creation of halo illustration. At the preprocessing stage, a view-independent halo volume is generated on graphics hardware based on the opacity volume, each voxel of which is an opacity mapped from the opacity transfer function. The halo volume captures all of the potential halos around the boundaries of features. During volume rendering, this halo volume is used to extract halos around the contours of features for the current viewpoint. In this way, halos can be rendered in real time and the performance of halo illustration is significantly improved.

The halo volume generation is based on the difference between the opacity volume and its Gaussian filtered volume. The efficiency of this depends largely on the size of the Gaussian kernel. Instead of using the fixed Gaussian kernel as in the previous work, this paper utilizes the data-dependent size of the Gaussian kernel derived from the estimated boundary thickness of the volume. As a result using the kernel with data dependent size, the resulting performance for halo generation is about twice as fast for the data sets investigated.

In order to further enhance depth perception, this paper also introduces a number of depth-dependent halo effects, e.g., depth color fading and depth width fading. The color and width of halos can be dynamically modulated based on the depth. These techniques provide the illustrator with more halo effects and perceptual cues for depth interpretation.

The remainder of the paper is structured as follows. After related work is reviewed in Sect. 2, an overview of our method is introduced in Sect. 3. Section 4 describes the generation of the view-independent halo volume, and we present halo visual mapping and depth-dependent halo effects in Sect. 5. Finally, we demonstrate the proposed method by applying it to several volume data sets in Sect. 6.

2 Related work

Illustrative visualization is an active research area in visualization, and numerous techniques have been developed to enhance the perception of shape, orientation, and depth information.

Illustrative visualization generally employs non-photorealistic rendering techniques that generate different rendering styles to convey structural information expressively, according to the circumstances. The gradient magnitude was first introduced by Levoy [9] to allow better boundary perception, while Rheingans and Ebert [14] presented various feature and orientation enhancements, such as boundary enhancement, silhouette enhancement, tone shading, distance color blending and distance color cues. Nagy et al. [11] added line drawings to direct volume rendering, as feature lines are one of the most effective visual abstractions for highlighting features of interest in traditional illustrations, and Kindlmann et al. [8] used curvature information to control the thickness of contours and emphasize ridges and valleys. The handcrafted shading techniques by Lu et al. [10] and Yuan and Chen [25] describe stippling and hatching on the surface of features to strengthen shape perception. Veronika et al. [22] suggested an illustrative shadow representation, which compensates the lower luminance range by adding a chromatic component to prevent overdarking. These illustrative shadows provide effective relative depth and surface cues for important structures in shadow.

Svakhine et al. [17] discussed illustration-inspired effective outlining techniques and selective depth enhancement, which are based on depth buffer unsharp masking to provide perceptual cues of object importance and spatial relationships. Volumetric unsharp masking proposed by Tao et al. [18] enhances the local contrast of features. They precomputed a smoothed view-dependent radiance volume and employed it to dynamically modulate the radiance of each sample during volume rendering. Although our method also calculates the halo volume in the preprocessing step, the halo volume is view-independent and the halo effects are more flexible than in their radiance enhancement. Similarly, Ruiz et al. [15] applied unsharp masking to the intensity volume to produce different illustrative effects, such as stipple rendering. While, in their approach, the importance of a feature is directly dependent on the original intensity value, the proposed opacity-volume based method allows the user to determine the importance of the features through the opacity transfer function, which is more flexible for the user.

Besides the low-level techniques above, there exist high-level techniques based on the focus + context principle to depict features expressively, and these techniques can be also seen as smart visibility techniques [20]. The magic volume lens proposed by Wang et al. [23] magnifies features of interest while suppressing the remaining volume regions. Importance-driven volume rendering introduced by Viola et al. [21] defines an object importance for each part of the volume, and the features with greatest importance can be made more clearly visible by reducing the opacity of features in front of them. Bruckner and Gröller [2] presented exploded views based on a force-based model to represent the focus feature clearly.

Several techniques have been developed to improve depth perception using halos. The haloed line effect proposed by Appel et al. [1] is used to convey the relative depth of lines. Interrante and Grosch [6] implemented 3D visibility-impeding halos in 3D flow visualizations using line integral convolution to clarify depth relationships. They created a halo volume containing slightly larger scan-converted spots at the preprocessing stage. Wenger et al. [24] employed a similar idea based on filtering for interactive volume rendering of thin thread structures. Although the proposed halo volume is also based on filtering, we can identify 3D boundaries of features to generate halos around both strong and weak structures. Everts et al. [5] used view-oriented triangle strips to create depth-dependent halos around lines. Their approach emphasizes tightly bounded lines and abstracts less structured lines.

In volume visualization, Rheingans and Ebert [14] introduced a halo volume based on the gradient vector and the viewing direction to generate feature halos. Later, Svakhine and Ebert [16] extended this work by moving the computation of the halo volume to graphics hardware for interactive volume illustration. This view-dependent halo volume needs a time-consuming recalculation for each viewpoint change as mentioned above, and the halo effect is also not very flexible. Bruckner and Gröller [3] presented volumetric halos based on screen-space halo generation to enhance depth perception. Their approach performs halo seeding, halo spreading, and halo rendering during volume rendering. In contrast to this, our method generates a halo volume in volume space at the preprocessing stage, in which most of the work in halo seeding and halo spreading of volumetric halos is performed. As a result, our approach can achieve better performance in halo rendering. In addition, we simplify the halo seeding procedure and further utilize depth-dependent halos to provide more effects to assist with depth perception.

3 Method overview

Our approach was originally inspired by high-dynamic-range (HDR) tone mapping [12], which reduces the contrast while preserving details for displaying HDR images on the low-dynamic-range media. This is usually based on a two-scale decomposition of the image into a base layer and a detail layer.

When a Gaussian low-pass filter is applied to an edge-like signal, it will create a strong peak and valley in the detail signal that is defined as the difference between the signal and the filtered signal, as illustrated in Fig. 1(a). Such a strong peak and valley produce a halo artifact in HDR tone mapping [4]. In this paper, we utilize the valley of the detail signal to generate halos around features of interest for volume illustration. As these halos depend only on the structures of features, we can generate them at the preprocessing stage and preserve them in a halo volume for real-time halo rendering.

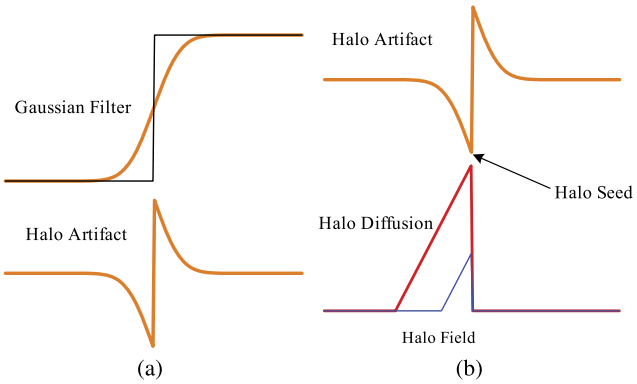


Fig. 1 The illustration of the halo field generation. (**a**) An edge-like signal is smoothed by a Gaussian low-pass filter and the detail signal, i.e., the difference between the signal and the smoothed signal, contains a strong peak and valley, which is the halo artifact. (**b**) The lowest valley is identified as the halo seed and the halo field is generated by halo diffusion in the low-value region outside the edge. Two different halo fields are illustrated in *red* and *blue*, respectively

A straightforward way to find all of the potential halos is to smooth the intensity volume and record the negative differences as the halo intensities. However, as features in the volume are usually defined by the transfer function, for features with gradual changing boundaries, the exact boundaries may be different from those identified directly from the intensity volume. In addition, halos may be in the high-scalar region for low-scalar features inside the high-scalar region, but halos identified from the scalar volume can lie only in the low-scalar region. As features of interest are usually assigned a higher opacity in the classification process, the importance and boundaries of features are well defined by the user-specified opacity. Thus, to create potential halos, instead of the original intensity volume, we make use of the opacity volume, which implicitly defines features of interest. Our approach can be easily extended to segmented features, as the boundaries are explicitly defined for segmented features.

An overview of our approach is shown in Fig. 2. At the preprocessing stage, described in detail in Sect. 4, the detail opacity volume is obtained by subtracting the smoothed opacity volume from the opacity volume, and halo seeds on the boundaries of features are extracted from the negative parts of the detail opacity volume. This step is called halo seeding. The halo seeds are then diffused to low-opacity regions outside the boundaries to generate a halo-intensity field, and these halo intensities are stored in a halo volume. The halo volume will need to be recalculated only if the opacity transfer function is modified.

At the rendering stage, the details of which are provided in Sect. 5, we render halos only around the contours of fea-

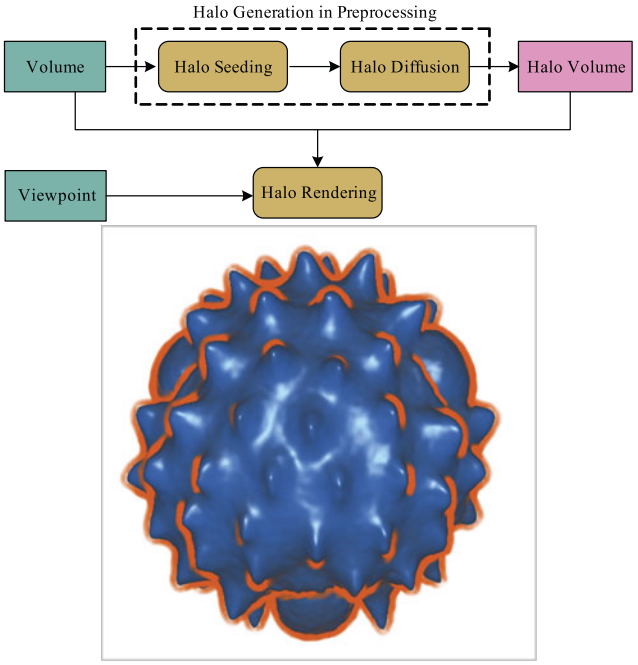


Fig. 2 The pipeline of halo generation and rendering. Halo generation involves two main steps: halo seeding and halo diffusion, and this preprocessing process creates a view-independent halo volume that contains all of the potential halos. The halo rendering uses the halo volume generated to create halos around the contours of structures for the current viewpoint during the volume rendering

tures that require knowledge of the viewing direction. The halos are integrated with the visual properties of the sample during volume rendering.

4 Halo volume generation

As illustrated in Fig. 2, halo volume generation consists of two steps: halo seeding and halo diffusion, which are described in the sections below.

4.1 Halo seeding

As features in the volume are defined by the transfer function, and features of interest are usually assigned with high opacity values, we make use of the opacity value to identify boundaries of user-defined features, i.e., the halo seeds that emit halos.

At the outset, the scalar value of each voxel is mapped to the opacity value via the opacity transfer function, and this produces an *opacity volume*. A Gaussian low-pass filter is then applied to the opacity volume to obtain a *smoothed opacity volume*. Finally, the opacity difference of each voxel is calculated and stored in the *detail opacity volume*. As illustrated in Fig. 1(b), the desired halos are located in the negative parts of the detail opacity volume.

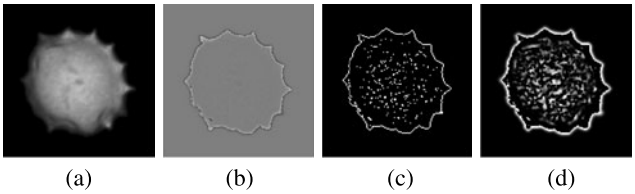


Fig. 3 The intermediate results of the halo volume generation. (**a**) A scalar slice. (**b**) The differences between (**a**) and its smoothed version. The values are increased by 0.5 to illustrate the negative parts. (**c**) Halo seeds. (**d**) Halo field

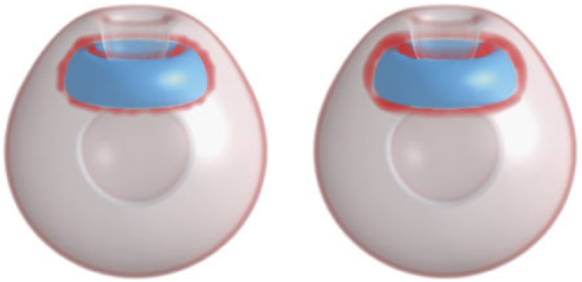
The Gaussian parameter σ has very little influence in this step. The only requirement is that the width of the Gaussian kernel should be larger than the thickness of the boundary to ensure that only one halo exists around each boundary. In volume visualization, it is usually assumed that real boundaries have a sharp and discontinuous change in the measured physical property, but that due to the band-limited property of scanning devices, measured boundaries are blurred by a Gaussian function, which is constant over a volume data set. The thickness of boundaries in a data set can be approximately estimated based on the work of Kindlmann and Durkin [7] as follows:

$$t = \frac{4\max(f'(x))}{\sqrt{e}(\max(f''(x)) - \min(f''(x)))}, \tag{1}$$

where $f'(x)$ and $f''(x)$ are the first and second directional derivatives along the gradient direction in the data set. With the recovered thickness, we can adaptively determine the optimal Gaussian parameter for each data set.

It would be possible to utilize the detail opacity volume directly as the halo volume used in the rendering process. However, the halos generated would be nonuniform as a result of the uneven opacity differences on the boundaries of different features; this would be particularly true for weak features. To address this problem, we explicitly identify the boundaries of features as halo seeds by checking the neighboring opacity differences for each voxel, for example, by use of a 6-point neighborhood. If any positive neighbors are found around the voxel, the voxel is viewed as a halo seed. These identified halo seeds are near the lowest points of valleys, as shown in Fig. 1(b). In this way, the boundaries of features defined by the opacity transfer function are extracted as halo seeds, regardless of the magnitudes of the opacity differences.

Figure 3 shows some intermediate results of halo volume generation for the daisy pollen grain data set. Figure 3(a) is one slice of the scalar volume. The halo artifact can be clearly observed in Fig. 3(b), in which the difference is increased by 0.5 to illustrate the negative parts. As can be seen from Fig. 3(c), halo seeds are accurately identified on the boundaries of features.



(a) (b)

Fig. 4 The smoothing effect: (**a**) Halos generated from original halo volume. (**b**) Halos generated from smoothed halo volume

4.2 Halo diffusion

The halo seeds are located on the boundaries of features with a width of one voxel. These halo seeds emit halos to the region outside the boundaries, and generate a halo intensity volume. This can be achieved by diffusing the halo intensity from these halo-seed voxels to neighboring voxels.

For each voxel, the distance d to the closest halo seed is computed up to a width of N voxels, where the user-defined parameter N, in the range $[1, 4]$, is the width of the required halos in terms of voxels . The halo intensity of each voxel is recorded as $\max(N - d, 0)$. Thus, the halo intensities near boundaries are larger than those in other regions, as illustrated in Fig. 1(b). To restrict the diffusion area to the exterior of structures, we do not calculate the distance for the voxels with positive differences during the diffusion process. Figure 3(d) shows the halo intensity field diffused from the halo seeds in Fig. 3(c).

The generated halo volume can be used directly in halo rendering. However, it can be seen clearly from Fig. 4(a) that the boundary of the halos generated is jagged, so we further smooth the halo volume to generate a smooth edge for the halos, as shown in Fig. 4(b). A simple Gaussian filter is used in our implementation, and this smoothing is also only applied to the negative parts of the detail opacity volume. The filtered halo volume is our final view-independent halo volume used for halo rendering.

4.3 GPU implementation

The generation of the halo volume can be implemented on graphics hardware, based on modern GPU features such as programmable shaders, frame-buffer objects, and render to 3D texture. This process requires two additional 3D textures. One floating-point texture is used to store the detail opacity volume and the float halo volume, and the other 8-bit texture serves as the intermediate halo volume.

The detail opacity volume is produced in one pass. The detail shader reads from one slice of the original volume, maps the scalar value to the opacity via the opacity transfer function, then averages opacities of neighbors based on the Gaussian filter. Finally, it calculates the difference, and writes the result to a 3D float texture slice. The identification of the boundary takes the detail opacity volume as input and outputs the seed mask to the intermediate halo volume. The halo field is diffused using a Dijkstra flood-fill algorithm in multiple passes. As the read and write positions in the halo diffusion process are different, we can use the intermediate halo volume as the input and output volume. The halo volume smoothing step uses the intermediate halo volume to generate the float halo volume.

From the discussion above, the view-independent halo volume requires recalculation only if either the opacity transfer function or the diffusion parameter N are changed. The halo volume can be dynamically generated on the GPU at interactive frame rates.

5 Halo rendering

The halo volume generated at the preprocessing stage is view-independent, and halos are formed around the boundaries of features. Thus, the rendering process integrates the halo volume with the viewpoint to draw halos only around the contours of features. The intensities of halos are mapped to the color and opacity value to yield various halo effects. Halo rendering is combined with direct volume rendering based on ray casting.

5.1 Halo mapping

During the ray casting, the view-independent halo intensity $h(p)$ at the sample position p can be interpolated from the halo volume. The halo intensity is in the internal range $[0, N]$ and is linearly remapped from a user-specified interval $[h_{\min}, h_{\max}]$ to $[0, 1]$ as follows:

$$h'(p) = \begin{cases} 0, & \text{if } h(p) < h_{\min}, \\ \frac{h(p)-h_{\min}}{h_{\max}-h_{\min}}, & \text{if } h_{\min} \le h(p) \le h_{\max}, \\ 1, & \text{if } h(p) > h_{\max}, \end{cases} \tag{2}$$

where $h'(p)$ is the normalized halo intensity, $h_{\max}$ allows users to define the start position of halos around the contours of structures, and $h_{\min}$ determines the width of halos. These parameters can be intuitively adjusted to create a user-desired halo effect at run time. Figures 5(a) and 5(b) show an example of the use of $h_{\min}$ to change the width of halos.

The normalized halo intensity $h'(p)$ is modulated with the dot product between the viewing direction $\hat{v}$ and the normalized gradient direction $\hat{n}_p$ to obtain the view-dependent halo intensity $h_v(p)$:

$$h_v(p) = h'(p) \cdot (1 - \hat{n}_p \cdot \hat{v})^{\alpha}, \tag{3}$$

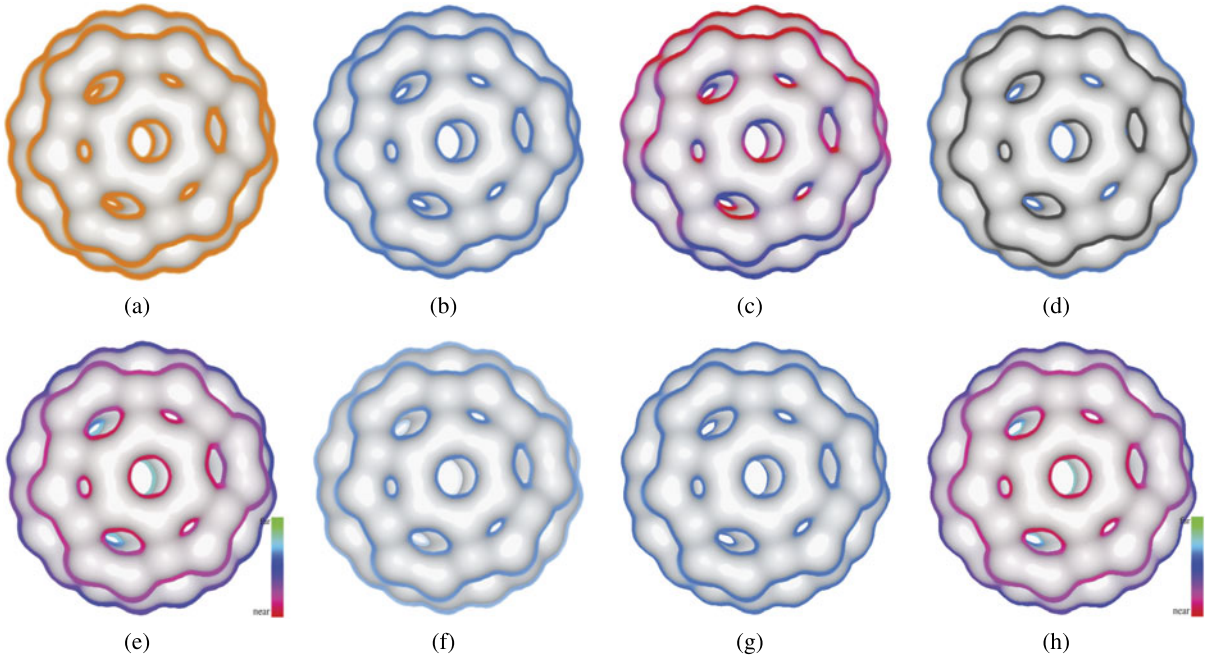


Fig. 5 Different halo effects for a simple volume C_{60}. The opacity of halos is modulated by the halo intensity. (**a**) Halos with a constant *orange* color. (**b**) Halos with a constant *blue* color. As the value of $h_{\min}$ is larger than the value in (**a**), the halo width is reduced. (**c**) Direction color mapping. The orientation of features from up to down is linearly mapped to the color from *red* to *blue*. (**d**) Occlusive halo. Halos that occlude other features are in the *black* color, similar to a shadow. (**e**) Depth color mapping. The depth of halos from near to far is linearly mapped to *red–blue–green* color. (**f**) Depth color fading. (**g**) Depth width fading. (**h**) The hybrid effect of depth color mapping and depth width fading

where the parameter α is used to control the precise location of halos on the contours of structures. To fix halos on the contours, a large number should be assigned to α, typically 8–32. The halo mapping is based on the view-dependent halo intensity $h_v(p)$, and a zero value means no halo effect.

We utilize the concept of a halo profile function [3] to map the view-dependent halo intensity to the color and opacity value, such as halo-modulated opacities with a constant color, and halo-modulated colors with a constant opacity. While a constant color is assigned to the whole halo in Figs. 5(a) and 5(b), the local orientation of features can be mapped to the color of halos in Fig. 5(c). In contrast to *emissive halos*, which emit light, there is another kind of halo, the *occlusive halo* [3], which absorbs light if they occlude other features. While Fig. 5(a) shows emissive halos, which emit orange light, Fig. 5(d) are the combination of emissive halos and occlusive halos, as the nonocclusive halos emit blue light and the occlusive halos absorb light behind the halos (in the dark color). Further effects will be introduced in the next section.

The mapping of the view-dependent halo intensity is performed at each sample during ray casting. The halo contribution of each sample is integrated with its original contribution, and these integrated contributions are blended in front-to-back order along the ray.

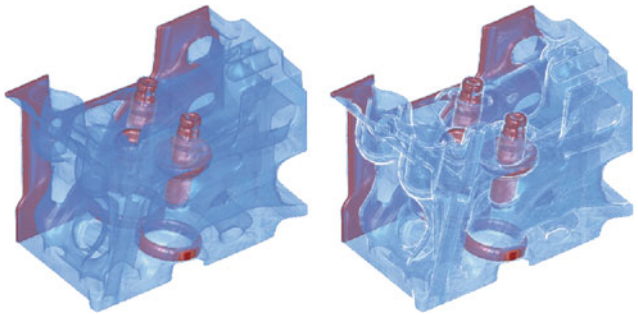
5.2 Depth-dependent halo effects

In order to further improve depth perception, we present several depth-dependent halo effects. The color and width of halos are dynamically modulated based on the depth information. In this paper, the depth value $d(p)$ at the sample p is defined as the distance from the sample to the nearest point of the volume from the viewpoint. To achieve various depth-dependent halo effects, we linearly remap the depth value $d(p)$ from a user specified interval $[d_{\min}, d_{\max}]$ to $[0, 1]$, as performed in Eq. (2). Here, $d_{\min}$ is the depth value at which to start the depth-dependent halo enhancement, and $d_{\max}$ is the depth value at which to end it, typically the largest depth value from the current viewpoint.

The color of halos at each sample can be modulated using the normalized depth value $d'(p)$. The normalized depth value $d'(p)$ can be mapped to the color of halos, similar to the view-dependent halo intensity. This effect is referred as depth color mapping. Another color effect is depth color fading, similar to distance color blending [16]. The normalized depth value $d'(p)$ is used to blend the original halo color $C_{\text{halo}}(p)$ and the background color $C_{\text{background}}$:

$$C(p) = \big(1 - d'(p)\big) \cdot C_{\text{halo}}(p) + d'(p) \cdot C_{\text{background}}. \qquad (4)$$

In this way, the color of rendered halos gradually fades as they recede from the viewpoint, similar to the effect of aerial perspective.

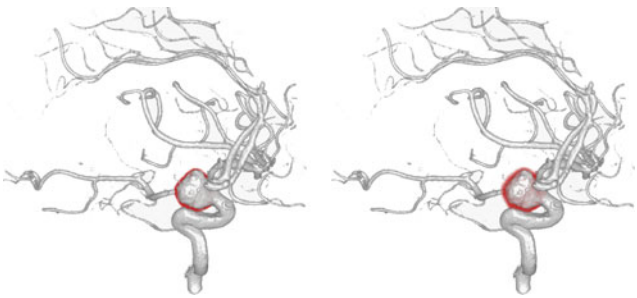


(a) (b)

Fig. 6 The effect of adding perceptual cues of spatial relationships to transparent structures. (**a**) The rendered result without halos. (**b**) The rendered result with a *white* halo

(a) (b)

Fig. 7 Halos are drawn only around the contours of bones to demonstrate the value-dependent effect. (**a**) The rendered result without halos. (**b**) The rendered result with a *blue* halo to separate the boundaries of bones



(a) (b)

Fig. 8 Halos are drawn only around the contours of the aneurysm to demonstrate the position-dependent effect. (**a**) A position-dependent halo is drawn around the contours of the aneurysm. (**b**) A position-dependent shell is drawn around the contours of the aneurysm

The width of halos can be also reduced along the viewing direction to enhance depth perception by resembling the perspective effect. We call this effect depth width fading. The constant parameter $h_{\min}$ is replaced by the position-dependent variable $h'_{\min}(p) = h_{\min} + (h_{\max} - h_{\min}) \cdot d'(p)$ during ray casting. Thus, the thickness of halos gradually decreases, as the depth value increases along the viewing direction.

Figure 5 shows some effects on a simple volume. The effect of depth color mapping is shown in Fig. 5(e), and the user can dynamically adjust the color range for mapping. Figure 5(f) shows the effect of depth color fading, and in Fig. 5(g), the width of halos is trimmed along the viewing direction. These depth-based halo effects can be combined to provide more perceptual cues for depth interpretation—Fig. 5(h) gives an example for the hybrid effect of depth color mapping and depth width fading.

6 Results and discussion

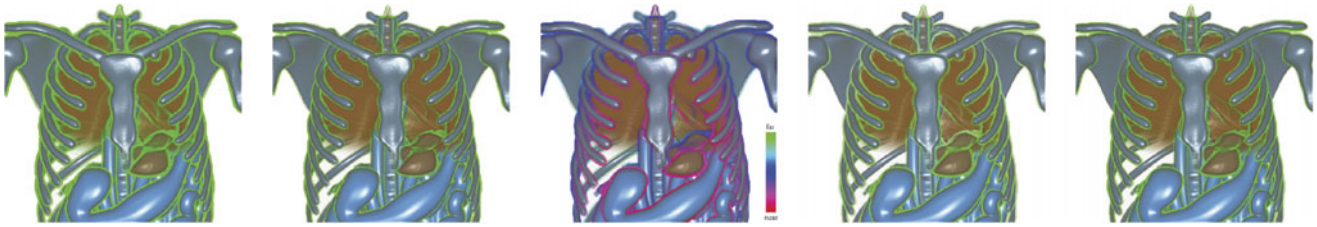
We have implemented the halo illustration technique discussed above in a GPU-based volume ray-caster to enhance depth perception for structures with complex depths.

Figure 2 is the daisy pollen grain data set with halo-modulated opacities and a constant color. The wavy local features can be easily separated from the base shape. Figure 6 shows the effect of adding perceptual cues of spatial relationships to transparent structures. Here, the inner structures of the engine data set are enhanced with a white halo. Compared to the result from standard direct volume rendering (DVR) in Fig. 6(a), the halo illustration provides clearer spatial relationships of inner structures.

Halos can be generated only around features with the specified scalar values based on the value influence function. The value influence function is defined in the 1D range space, like the transfer function based on the scalar value. Figure 7 displays an example of such an effect. These tiny halos are formed around the contours of bones only and make different bones easy to separate visually.

Halos can also only be located around features of interest using the positional influence function, as shown in Fig. 8. The positional influence function is a spherical focus region, and the focus point is defined by clicking on the slice. A red halo is drawn around the contours of the aneurysm as illustrated in Fig. 8(a). This effect can effectively attract the viewer's attention to the aneurysm, which is common in traditional illustrations. We can also render halos around the boundaries of a structure, not just the contours. This provides a fog effect using the halo volume, as displayed in Fig. 8(b). Using these feature-dependent halo effects, important features can be selectively emphasized by rendering halos only around them.

The halo effect can be adjusted by the parameters $h_{\min}$ and $h_{\max}$, as they determine the width and start position of halos. Compared to the original halos in Fig. 9(a), the width of halos is reduced for a larger value of $h_{\min}$ in Fig. 9(b). Volumetric unsharping masking [18] provides similar halo results on this data set. However, it only makes the boundary of features lighter or darker. Our halo effect is more flexible and expressive. Figure 9(c) shows the effect of depth



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

Fig. 9 The comparison of different halo effects. (**a**) Halo illustration. (**b**) Halo illustration with a larger value $h_{\min}$, for comparison. (**c**) Depth color mapping. (**d**) Depth color fading. (**e**) Depth width fading

Table 1 Performance comparison between DVR and halo rendering (frames/s)

Data set	Size	DVR	Halo rendering	(%)
Pollen	$192 \times 180 \times 168$	60.31	48.49	80.4
Engine	$256 \times 256 \times 128$	53.21	34.09	64.1
Foot	$256 \times 256 \times 250$	52.98	28.55	53.9
Phantom	$256 \times 256 \times 256$	51.21	28.18	55.0

color mapping, with the same depth being mapped to the same color to add more depth cues. The effects of depth color fading and depth width fading are shown in Figs. 9(d) and 9(e), respectively. Such depth-dependent halo effects provide more perceptional cues for depth understanding than simple volumetric halos.

The performance of halo rendering compared to DVR and the size of the data sets in our experiments are listed in Table 1. The data sets are illustrated in Figs. 2, 6, 7, and 9. The performances were measured on an Intel Core 2 Quad Q9550 (2.83 GHz) processor with an NVIDIA GeForce GTX 285. The sampling distance for the ray casting was 0.5 of the voxel size, and the viewport size was 512×512. The frame rate was averaged over a 360 degree rotation along each axis. These rendering parameters are almost the same as the parameters used for performance comparison in [3].

Halo rendering takes place at more than half of the frame rate of DVR. The proposed object-space halo illustration is therefore faster than screen-space halo illustration, which achieves only one third of the frame rate of DVR even after sacrificing the depth accuracy [3]. The main reason is that our approach generates a view-independent halo volume at the preprocessing stage, which includes most of the work involved in halo generation, while screen-space halo illustration performs this work during the volume rendering.

The halo volume generation took 0.075 s and 0.143 s for the engine and phantom data sets, respectively, with a Gaussian kernel of 3^3 and a diffusion parameter N of 2. As the estimated boundary thicknesses of the engine and phantom are roughly 0.86 and 1.25 voxels, the size of the Gaussian kernel 3^3 is large enough for halo seeding. Halo volume generation using the data-dependent Gaussian kernel based on the boundary thickness is almost twice as fast compared to the fixed Gaussian kernel $\sigma = 1.0$ used in the previous work [19]. Although this step is time-consuming for large volumes, it still allows the user to adjust the opacity transfer function at interactive frame rates. The most computationally-intensive operation is the Gaussian filter, as this requires a large number of texture fetches on graphics hardware. Illustrative visualization is usually applied to the volume after the features of interest have been classified by the transfer function, so the opacity transfer function is not changed as frequently as the viewpoint during illustrative visualization.

One limitation of our approach is the large storage requirement associated with the two additional 3D textures in the halo volume generation. As the size of the volume grows, it becomes increasingly memory-consuming to store the full-resolution halo volume. In most applications, the performance improvement using object-space halo generation is worth the additional texture memory. It is fortunate that the on-board memory of GPUs continues to increase to satisfy the requirements of memory-hungry applications.

Another limitation is that the quality of the halo effect depends on the classification, especially the opacity transfer function, as the halo seeds are obtained through the opacity volume. If the features of interest are not well defined by the opacity transfer function, as in the case of noisy features, it will tend to introduce halo artifacts in the rendered image.

7 Conclusion

In this paper, we have presented a simple and effective method to create halo illustration. A view-independent halo volume based on the opacity volume generated at the preprocessing stage is used to accelerate halo rendering in the rendering process. Compared to screen-space halo illustration [3], the proposed object-space halo illustration simplifies the halo seeding procedure that directly depends on the feature classification, and it also greatly improves the performance of halo rendering using the precomputed halo volume.

We have also proposed depth-dependent halo effects, which dynamically adjust the color and width of generated halos based on the depth. Such halo effects further strengthen depth perception of complex structures and provide a clear sensation of spatial relationships. In the future, it would be interesting to perform a user study to quantify the depth perceptual gains of halo effects in volume visualization.

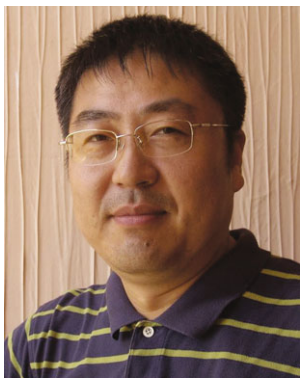
Acknowledgements This work was partially supported by 863 Program Project 2012AA12A404, NFS of China (No. 60873122 and No. 60903133), and the Open Project Program of the State Key Lab of CAD&CG (Grant No. A1012), Zhejiang University. The data sets are courtesy of Olaf Ronneberger, SFB 382, General Electric, Philips Research, VoreenPub.

References

1. Appel, A., Rohlf, F.J., Stein, A.J.: The haloed line effect for hidden line elimination. In: Proceedings of ACM SIGGRAPH '79, pp. 151–157 (1979)
2. Bruckner, S., Gröller, M.E.: Exploded views for volume data. IEEE Trans. Vis. Comput. Graph. **12**(5), 1077–1084 (2006)
3. Bruckner, S., Gröller, M.E.: Enhancing depth-perception with flexible volumetric halos. IEEE Trans. Vis. Comput. Graph. **13**(6), 1344–1351 (2007)
4. Chiu, K., Herf, M., Shirley, P., Swamy, S., Wang, C., Zimmerman, K.: Spatially nonuniform scaling functions for high contrast images. In: Proceedings of Graphics Interface '93, pp. 245–253 (1993)
5. Everts, M.H., Bekker, H., Roerdink, J.B., Isenberg, T.: Depth-dependent halos: illustrative rendering of dense line data. IEEE Trans. Vis. Comput. Graph. **12**(6), 1299–1306 (2009)
6. Interrante, V., Grosch, C.: Strategies for effectively visualizing 3D flow with volume LIC. In: Proceedings of Visualization'97, pp. 421–424 (1997)
7. Kindlmann, G., Durkin, J.W.: Semi-automatic generation of transfer functions for direct volume rendering. In: Proceedings of the 1998 IEEE Symposium on Volume Visualization (VVS'98), pp. 79–86. ACM, New York (1998)
8. Kindlmann, G., Whitaker, R., Tasdizen, T., Möller, T.: Curvature-based transfer functions for direct volume rendering: methods and applications. In: Proceedings of Visualization '03, pp. 513–520 (2003)
9. Levoy, M.: Display of surfaces from volume data. IEEE Comput. Graph. Appl. **8**(3), 29–37 (1988)
10. Lu, A., Morris, C.J., Taylor, J., Ebert, D., Hansen, C., Rheingans, P., Hartner, M.: Illustrative interactive stipple rendering. IEEE Trans. Vis. Comput. Graph. **9**(2), 127–138 (2003)
11. Nagy, Z., Schneider, J., Westermann, R.: Interactive volume illustration. In: Proceedings of Vision, Modeling and Visualization, pp. 497–504 (2002)
12. Paris, S., Kornprobst, P., Tumblin, J., Durand, F.: A gentle introduction to bilateral filtering and its applications. In: ACM SIGGRAPH 2007 Course Notes (2007)
13. Rautek, P., Bruckner, S., Gröller, M.E., Viola, I.: Illustrative visualization—new technology or useless tautology? Comput. Graph. Q. **42**(3) (2008). http://www.siggraph.org/publications/newsletter/volume-42-number-3/illustrative-visualization-2013-new-technology-or-useless-tautology
14. Rheingans, P., Ebert, D.S.: Volume illustration: nonphotorealistic rendering of volume models. IEEE Trans. Vis. Comput. Graph. **7**(3), 195–202 (2001)
15. Ruiz, M., Boada, I., Feixas, M., Sbert, M.: Interactive volume illustration using intensity filtering. In: Proceedings of Computational Aesthetics in Graphics, Visualization, and Imaging 2010, pp. 51–58 (2010)
16. Svakhine, N.A., Ebert, D.S.: Interactive volume illustration and feature halos. In: Proceedings of Pacific Graphics '03, pp. 347–354 (2003)
17. Svakhine, N.A., Ebert, D.S., Andrews, W.M.: Illustration-inspired depth enhanced volumetric medical visualization. IEEE Comput. Graph. Appl. **15**(1), 77–86 (2009)
18. Tao, Y.B., Lin, H., Bao, H.J., Dong, F., Clapworthy, G.: Feature enhancement by volumetric unsharp masking. Vis. Comput. **25**(5–7), 581–588 (2009)
19. Tao, Y.B., Lin, H., Dong, F., Clapworthy, G.: Opacity volume based halo generation for enhancing depth perception. In: Proceedings of the International Conference on CAD/Graphics 2011, pp. 418–422 (2011)
20. Viola, I., Gröller, M.E.: Smart visibility in visualization. In: Proceedings of Computational Aesthetics in Graphics, Visualization and Imaging '05, pp. 209–216 (2005)
21. Viola, I., Kanitsar, A., Gröller, M.E.: Importance-driven feature enhancement in volume visualization. IEEE Trans. Vis. Comput. Graph. **11**(4), 408–418 (2005)
22. Šoltészová, V., Patel, D., Viola, I.: Chromatic shadows for improved perception. In: Proceedings of Non-Photorealistic Animation and Rendering (NPAR 2011), pp. 105–115. ACM, Vancouver (2011)
23. Wang, L., Zhao, Y., Mueller, K., Kaufman, A.: The magic volume lens: an interactive focus+context technique for volume rendering. In: Proceedings of Visualization '05, pp. 47–54 (2005)
24. Wenger, A., Keefe, D.F., Zhang, S., Laidlaw, D.H.: Interactive volume rendering of thin thread structures within multivalued scientific datasets. IEEE Comput. Graph. Appl. **10**(6), 664–672 (2003)
25. Yuan, X., Chen, B.: Illustrating surfaces in volume. In: Proceedings of Joint IEEE/EG Symposium on Visualization '04, pp. 9–16 (2004)

Yubo Tao received the B.S. and Ph.D. degree in computer science and technology from Zhejiang University in China, in 2003 and 2009, respectively. He worked in the State Key Laboratory of CAD&CG of Zhejiang University as a Postdoctoral Researcher. Then he joined the Centre for Computer Graphics & Visualization (CCGV) at the University of Bedfordshire as a Research Fellow. He is currently an Assistant Researcher in the State Key Laboratory of CAD&CG of Zhejiang University. His research interests include scientific visualization and computational electromagnetics.

Chao Wang received the B.S. degree from Harbin Institute of Technology, China, in 2001, the M.Sc. degree from Nanjing University, China, in 2004, and the Ph.D. degree from Tsinghua University, China, in 2010. After that, he worked at the University of Hamburg as a senior member of CINACS. Then he joined the CCGV lab of the University of Bedfordshire as a research fellow. He is currently a research fellow of the University of Wollongong, Australia. His research interests include image processing and computer vision.



Hai Lin received the B.Sc., M.Sc., in electrical engineering from Xidian University, Xi'an, China, in 1987, 1990 respectively. He received the Ph.D. degree in Computer Science from Zhejiang University, Hangzhou, China in 2000. Currently, he is a Professor of Visual Computing in the State Key Lab of CAD&CG, Zhejiang University. He is also a visiting professor of the Department of Computing and Information Systems, University of Bedfordshire, UK. His research interests include computational electromagnetic, computer graphics, and scientific visualization.

Feng Dong is a professor of Visual Computing in the Department of Computing and Information Systems, University of Bedfordshire, UK. His research interests include fundamental computer graphics algorithms, texture synthesis, image-based rendering, medical visualization, volume rendering, human modeling/rendering, and virtual reality. He received a Ph.D. in Computer Science from Zhejiang University, China.

Gordon Clapworthy is Professor of Computer Graphics and Head of the Centre for Computer Graphics & Visualization (CCGV) at the University of Bedfordshire, Luton, UK. He received a B.Sc. (1st class) in Mathematics and a Ph.D. in Aeronautical Engineering from the University of London, and an M.Sc. (dist.) in Computer Science from the City University in London. He has published nearly 150 refereed papers in journals and international conferences and is a member of ACM, ACM SIGGRAPH and Eurographics. His areas of interest span computer graphics, computer animation, visualization, computer interaction, virtual reality, and biomechanics.