Combined Surface and Volumetric Occlusion Shading

Mathias Schott [*] University of Utah
Tobias Martin[†] University of Utah
A.V. Pascal Grosset[‡] University of Utah
Carson Brownlee[§] University of Utah
Thomas Höllt [¶] King Abdullah University of Science and Technology
Benjamin P. Brown [‖] University of Wisconsin
Sean T. Smith[**] University of Utah
Charles D. Hansen[††] University of Utah

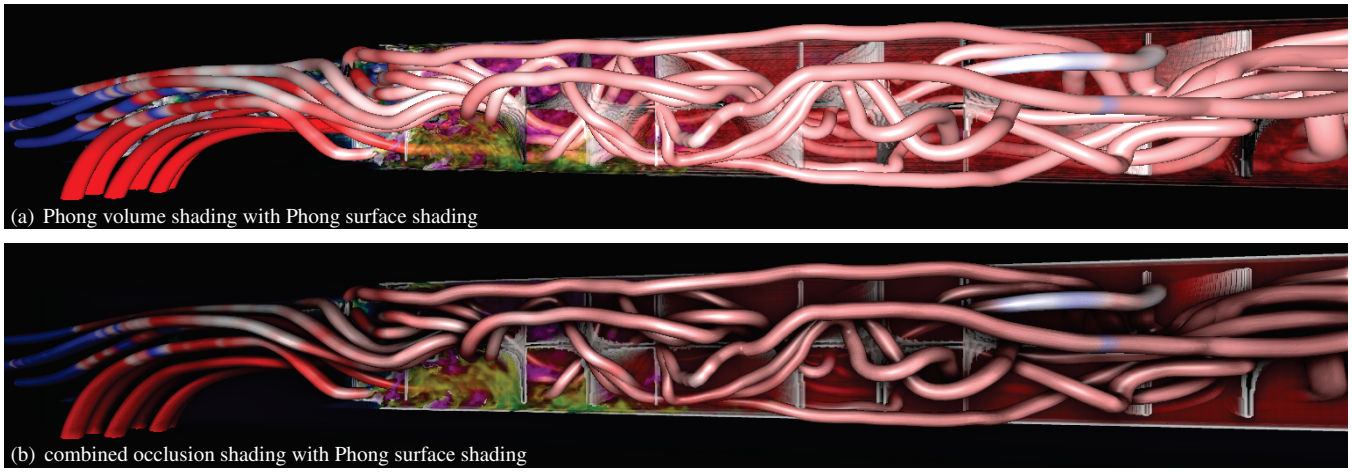

Figure 1: Stream lines (127k triangles) of a mixing pipe and a scalar volume ($1800 \times 121 \times 121$ resolution), of which the front half was removed by a clipping plane. The images were rendered with a) Phong volume shading with on-the-fly gradient estimation and Phong surface shading and b) the presented combined surface and volumetric occlusion shading method.

Abstract

In this paper, a method for interactive direct volume rendering is proposed that computes ambient occlusion effects for visualizations that combine both volumetric and geometric primitives, specifically tube shaped geometric objects representing streamlines, magnetic field lines or DTI fiber tracts. The proposed algorithm extends the recently proposed Directional Occlusion Shading model to allow the rendering of those geometric shapes in combination with a context providing 3D volume, considering mutual occlusion between structures represented by a volume or geometry.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism— Subjects: Color, shading, shadowing

1 Introduction

Data sets arising in many scientific research areas are often combinations of commonly used scalar fields and other types of data. Vector fields typically arise while simulating the mechanics of fluids or gases. A variety of methods exist to render those directly, but often, streamlines are traced through those vector fields in order to create a high level abstracted visualization of those vector fields. Similarly, DTI fiber tractography is used to gain insight about the configuration and orientation of neural pathways in the field of neurosciences in order to increase the understanding of the functioning of the brain.

It is desirable to combine the visualization of a scalar field with that of streamlines or DTI fibers since they are typically registered with respect to each other. This poses a problem however since the visual interplay between those rather different data modalities introduces additional context that is key in comprehending the combined data sets. It is especially important to be able to reason about the spatial arrangement of the geometry with respect to the volume. Existing visualization techniques are less suited to helping users gain insight into those data sets, since commonly available techniques employ only local information in the form of Phong shading.

In this paper, we present a method for interactive direct volume rendering that allows the computation of occlusion effects for volumetric data sets with both solid and transparent features, combined with solid tube-like geometry from DTI fiber tractography or streamline tracing. The proposed method is an extension of the directional occlusion shading model presented by Schott *et al.* [14] and as such is based on incremental filtering, which has been successfully used in the past to approximate integration for computing advanced volumetric scattering and shading effects [7, 11, 13, 14, 20].

The occlusion effects of the geometric structures are incorporated straightforwardly by modifying the occlusion buffer update of the volumetric occlusion shading method. Additionally, a vicinity occlusion term is computed using the depth buffer of the geometric structures in order to capture more detailed occlusion effects introduced by the geometry. The combined occlusion effects of both the geometry and the volume provide additional context by increasing the spatial comprehensibility due to the consideration of neighboring structures of both volumetric and geometric origin.

[*]e-mail:mschott@cs.utah.edu
[†]e-mail:martin.cs.utah.edu
[‡]e-mail:zetval@cs.utah.edu
[§]e-mail:brownlee@cs.utah.edu
[¶]email:thomas.hollt@kaust.edu.sa
[‖]email:bpbrown@astro.wisc.edu
[**]e-mail:sean.t.smith@utah.edu
[††]e-mail:hansen@cs.utah.edu

IEEE Pacific Visualization Symposium 2012
28 February - 2 March, Songdo, Korea
978-1-4673-0866-3/12/$31.00 ©2012 IEEE

2 Related Work

The directional occlusion shading method presented by Schott *et al.* [14] builds the foundation for the extension considering the mutual occlusion of volumetric and geometric structures. It computes volumetric occlusion effects by maintaining and blurring an additional occlusion buffer while traversing the slices. Screen Space Ambient Occlusion techniques have recently received considerable interest after details about commercial implementations were presented publicly [10]. They use the depth buffer of a scene in order to approximate the geometry in the neighborhood of a pixel and use this representation to estimate the occlusion, thus being independent of the geometric complexity of the scene. Shanmugam *et al.* [16] compute high and low frequency occlusion by defining spheres at each pixel within which they then sample the depth and normal buffers. Our proposed method also computes high and low frequency occlusion effects separately, but differs by capturing low frequency occlusion effects in the volumetric occlusion buffer, instead of approximating them with spheres. Horizon based ambient occlusion methods [1] use the tangent plane of the surface in order to avoid evaluating the occlusion in those parts of the hemisphere that are known to be occluded.The Alchemy screen space ambient obscurance method [8] combine aspects of previous screen space ambient occlusion methods with a different derivation of obscurance in order to increase robustness and performance of their algorithm, which captures obscurances from details of varying scale and provides artist control over key parameters of their model. Vicinity Occlusion Maps [3], a direct volume rendering method, use a similar approach to shade a depth buffer derived from accumulated opacities. Summed area tables are used to determine also the low frequency, global occlusion effects of surface like structures contained in the volumetric data set. In contrast, our proposed method uses directional occlusion shading to compute the occlusion effects for the volume and global geometric structures, and a vicinity occlusion term based on the depths of the geometry to determine the occlusion of the high frequency, local geometric structures.

Wenger *et al.* [21] employ a two level rendering method to combine scalar volume rendering with rendering of vector fields, represented by thin threads which are created by filtering the lines into volumes using a cubic B-spline filter spanning multiple voxels. The resolution of those derived thread and halo volumes limits the maximal representable thread radius. Attributes of threads and halos are stored in those volumes and transfer functions are used to classify those, allowing selective culling by setting opacity accordingly. Melek *et al.* [9] use self orienting surfaces [15] to render hundreds of thousands of threads interactively on the GPU. Depth cuing and local lighting are used to shade those surfaces. A non-interactive global illumination rendering method based on hair rendering is proposed to yield high quality images. There, opacity shadow maps are used to provide self shadowing of the threads. An ambient occlusion term for rectangular shaped volumes is derived, based on the distance to the boundary of the volume. Stoll *et al.* [17] use a combined CPU / GPU approach to splat lines as generalized cylinders with local lighting and halos and texture and shadow mapping. Tessellated geometry is rendered in areas where the splatted impostors are parallel to the view direction. Zöckler *et al.* [23] exploit fixed-function texture mapping hardware to compute Phong lighting on transparent streamline segments. Stompel *et al.* [18] render streamlines non-photo realistically as strokes of varying lengths. Color gradients and semi-transparency are used to indicate directional information. Interrante *et al.* [6] perform 3D Line Integral Convolution (LIC) to create a flow representing color texture. Variations in hue are used to visually separate dense line bundles, volumetric halos to increase depth perception. Li *et al.* [19] scan convert streamlines into a 3D texture as a preprocessing step, while extracting additional attributes. This allows an interactive exploration by manipulation of a dependent lookup into an *appearance texture*, which can be used to give the effect of illuminated streamtubes, tone shading, or silhouette enhancements. Everts *et al.* [5] use an illustrative approach to render dense line data by creating halos around line bundles, while also increasing the depth perception by attenuating their widths. Their method could be combined with scalar volume rendering, however it is unclear how to maintain context between the volume and the geometry.

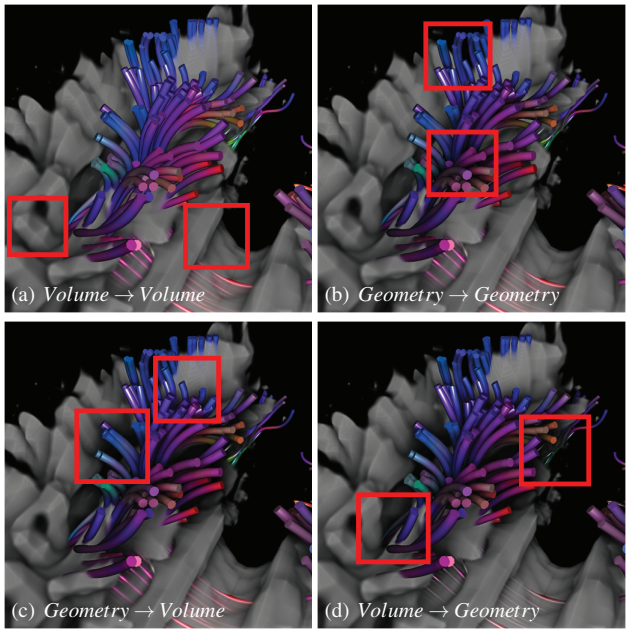

Figure 2: The various occlusion effects between the volume and geometry, where each image introduces additional interactions, beginning with the occlusion of a) the volume , b) the geometry, c) between the geometry and the volume and d) between the volume and the geometry.

3 Combining Occlusion Effects

An algorithm aiming to provide combined rendering of volumetric and geometric structures has to address their mutual interactions, and how to incorporate them into the overall rendering method. Volumetric shading methods that consider only basic local shading models have a minimal set of those interactions, such as determining the visibility of voxels behind the geometry in order to avoid computing their contributions, and properly attenuating the background during compositing the volume on top of the geometry. However, semi-global methods, such as the volumetric directional occlusion shading method [14], introduce additional interactions that need to be addressed in order to provide for a visualization technique that renders depth cues that plausibly combine the occlusion effects of both volumetric and geometric structures.

Volumetric occlusion effects from the *volume* into the *volume* can be computed using the directional occlusion shading approach [14], and can easily be combined with the Phong surface shading of the geometry, as Figure 2(a) shows. Notable here is the lack of depth discrimination between the geometric structures themselves. Occlusion effects between the geometric structures can be computed independently using a screen space ambient occlusion approach, as Figure 2(b) demonstrates, which enhances the depth perception of the geometry, due to the approximated occlusion effects. This naive method however considers each type of occlusion effects *independently*, thus missing important interactions by ignoring occlusion shadows cast from the *geometry* into the *volume* (Figure 2(c)) and those from the *volume* onto the *geometry* (Figure 2(d)). Only the

combination of all the occlusion effects between the volume and the geometry create a consistently shaded rendering which enhances the depth perception across the whole image, which is important in order to create a comprehensive visualization. The proposed combined method for rendering occlusion effects considers all the occlusion effects highlighted in Figure 2 and encompasses:

a) an adapted volumetric occlusion shading method that is modified to consider the shadows cast by the *geometry* into the *volume* in addition to those cast by the *volume* onto the *volume*, b) an additional data structure, the so called *partial occlusion buffer*, which stores the shadows cast by the *volume* onto the *geometry*, c) an additional screen space ambient occlusion term, the so called *vicinity occlusion term*, which computes the occlusion effects between *geometry* and surrounding *geometry*, based on inspiration by recently proposed techniques [1,3] and d) the combination of the volumetric occlusion term with the partial occlusion term yielding a combined occlusion term used to shade the geometric surfaces.

3.1 Integration into a Slice-Based Volume Renderer

Algorithm 1 The main combined occlusion shading algorithm

```
ComputeGBuffer()
eye_buffer ⇐ background_color
volume_occl_buf_next ⇐ volume_occl_buf_prev ⇐ 1
partial_occl_buffer ⇐ 1
for all slice s in all slices do
    UpdateEyeBuffer(s) {Algorithm 2}
    UpdatePartialOcclusionBuffer(s) {Algorithm 4}
    UpdateVolumetricOcclusionBuffer(s) {Algorithm 3}
end for
CompositeBuffers() {Algorithm 6}
```

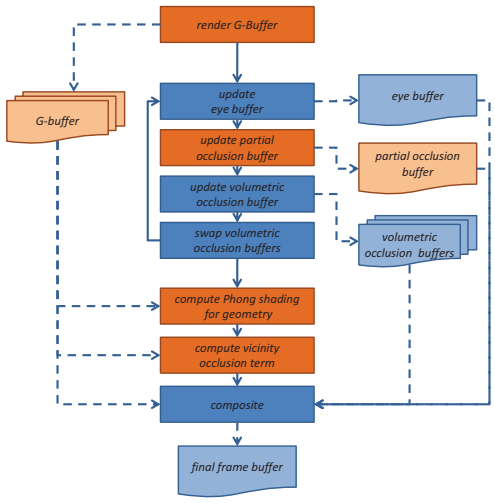

Figure 3: The extensions to the original DOS method are denoted in orange: additional data structures (G-buffer, partial occlusion buffer) and additional processing steps (computing the G-buffer, updating the partial occlusion buffer, computing surface shading and determining the vicinity occlusion term)

The extended algorithm, as outlined in Algorithm 1 and illustrated in Figure 3, is integrated into a slice-based volume renderer which computes volumetric occlusion effects [14].

The geometric structures, such as DTI fiber tracts or streamlines are rendered before the slice traversal to capture their colors, positions, normals and depth values into images. Those images, commonly referred to as a G-Buffer [12], are used to defer computation of surface lighting and the vicinity occlusion term until after the slice traversal. Deferring those computations is necessary in order to incorporate the volumetric occlusion term and the shaded volume into the final compositing step. As in the volumetric occlusion shading method, the volumetric occlusion buffer is used to store and compute the volumetric occlusion factor for each slice, additionally incorporating the occlusion from the geometry. An additional render pass then uses the volumetric occlusion buffer to update the partial occlusion buffer, storing the occlusion of the volume in front of the geometric structures. The eye buffer and the volumetric occlusion buffers get updated alternately during the slice-by-slice traversal of the volume. The view-aligned slices are computed on the CPU and then rendered from the eye's point of view, projecting them into the eye buffer, during which the volume is shaded based on the volumetric occlusion buffer and the result of the transfer function lookup. The volumetric occlusion information is updated by rendering the slice again, but now into the next volumetric occlusion buffer. During this pass, the occlusion factors of the previous volumetric occlusion buffer are integrated by reading and averaging multiple samples. The slice is rendered another time during which the volumetric occlusion buffer is read and used to update the partial occlusion buffer with the combined volumetric and low frequency geometric occlusion. The current and next occlusion buffers are swapped and the algorithm proceeds to the next slice, until all slices have been processed. Finally, the vicinity occlusion term for the geometric structures is computed and combined with the partial occlusion term and the G-Buffer during the final compositing.

3.1.1 Rendering the G-Buffer

Before traversing the slices, the geometry is rendered with a simple set of shaders that transform them and store their colors, view space positions, view space normals and view space distance in the individual 2D images comprising the G-Buffer. The stencil buffer is used to mask all the fragments covered by the streamlines geometry in order to optimize updating of the partial occlusion buffer. The data sets discussed in Section 4 are of static nature due to their origin; this however is not a limitation of the presented method, since one could easily stream time varying geometry or even perform streamline tracing on the graphics card.

3.1.2 Updating the Eye Buffer

Algorithm 2 UpdateEyeBuffer(s)

```
BindTexture(volume_occl_buf_prev, geom_linear_depth)
SetRenderTarget(eye_buffer, geom_depth_stencil)
EnableDepthTest(PASSIFLESS,DONOTWRITEDEPTH)
EnableBlending(ONEMINUSDESTINATIONALPHA,ONE)
for all fragments f of s do
    (x, |∇x|) ⇐ Texture3D(volume,f )
    (σ_s, σ_t) ⇐ Texture2D(transfer_function, (x, |∇x|))
    occlusion ⇐ Texture2D(volume_occl_buf_prev, f_clip)
    delta = |f_viewz − Texture2D(geom_linear_depth,f_clip)|
    sampling_distance = min(slice_distance, delta)
    α ⇐ 1 − e^(−σ_t · sampling_distance)
    frame_buffer ⇐ (L_a · σ_s · occlusion · α, α)
end for
```

The eye buffer update, as detailed in Algorithm 2, is performed similar to the volumetric occlusion shading method. The volume and transfer function are evaluated and the resulting color value is modulated by the current volumetric occlusion buffer, additionally testing the front-to-back composited slices against the depth-buffer.

3.1.3 Computing the Volumetric Occlusion Term

The occlusion factors of the previous volumetric occlusion buffer are integrated by reading and averaging multiple samples, as described in [14]. An additional look up into the geometric depth image is used to determine whether the current slice sample is behind the geometric structures, and if so, a user specified, (high)

Algorithm 3 UpdateVolumetricOcclusionBuffer(s)

```
BindTexture(volume_occl_buf_prev, geom_depth)
SetRenderTarget(volume_occl_buf_next)
DisableBlending()
for all fragments f of s do
    (x, |∇x|) ⇐ Texture3D(volume, f)
    σ_t ⇐ Texture2D(transfer_function, (x, |∇x|))
    if f_depth < Texture2D(geom_depth, f_clip) then
        σ_t ⇐ geometry_opacity
    end if
    occlusion ⇐ ComputeDOS(volume_occl_buf_prev, σ_t)
    frame_buffer ⇐ occlusion
end for
Swap(volume_occl_buf_prev, volume_occl_buf_next)
```

opacity value is used to *replace* the opacity resulting from the volume/transfer function lookup in order to incorporate the occlusion of the geometric structure into the volume. This pass, as detailed in Algorithm 3, is computing the occlusion effects from the *volume* into the *volume* and from the *geometry* into the *volume*.

3.1.4 Computing the Partial Occlusion Term

Algorithm 4 UpdatePartialOcclusionBuffer(s)

```
BindTexture(volume_occl_buf_prev, volume_occl_buf_next,
geom_linear_depth)
SetRenderTarget(partial_occl_buffer, geom_depth_stencil)
DisableBlending()
EnableDepthTest(PASSIFLESS, DONOTWRITEDEPTH)
EnableStencilTest() {update pixels that contain geometry}
StencilFunc(EQUAL, 1, ~0)
StencilOp(KEEP, DECREASE, KEEP)
for all fragments f of s − 1 do
    z_surface = Texture2D(geom_linear_depth, f_clip)
    z_slice = f_viewz, z_Δ = |z_surface − z_slice|
    z_ratio = min(slice_distance, z_Δ) / slice_distance
    occl_prev ⇐ Texture2D(volume_occl_buf_prev, f_clip)
    occl_next ⇐ Texture2D(volume_occl_buf_next, f_clip)
    partial_occlusion = occl_prev · (1 − z_ratio) + occl_next · z_ratio
    frame_buffer ⇐ partial_occlusion
end for
```

The partial occlusion buffer is used to record the volumetric occlusion from the beginning of the volume up to the surface, as Figure 4 and Algorithm 4 illustrate. It is computed by interpolating between the volumetric occlusion $volumetric\_occlusion_n$ of the current slice and the volumetric occlusion $volumetric\_occlusion_{n-1}$ of the previous slice, in order to approximate soft shadows cast onto the surface at distance $z_{surface}$ in view space.

The *previous* slice $slice_{n-1}$ is rendered instead of the current slice $slice_n$ because the latter will be discarded by the depth test due to it being occluded by the surface. The interpolation factor z_{ratio} is determined by the difference z_Δ between the surface and the slice, as shown in Equations 1 and 2. The difference z_Δ is clamped by the slice distance d in order to correctly handle the case where there is no surface between two subsequent slices. The $partial\_occlusion_n$ of the current $slice_n$, is then computed as outlined in Equation 3:

$$z_\Delta = |z_{slice} - z_{surface}| \quad (1)$$

$$z_{ratio} = \frac{\min(z_\Delta, d)}{d} \quad (2)$$

$$partial\_occlusion_n = volumetric\_occlusion_{n-1} \cdot (1 - z_{ratio}) + volumetric\_occlusion_n \cdot z_{ratio} \quad (3)$$

The stencil buffer mask, which was created while computing the

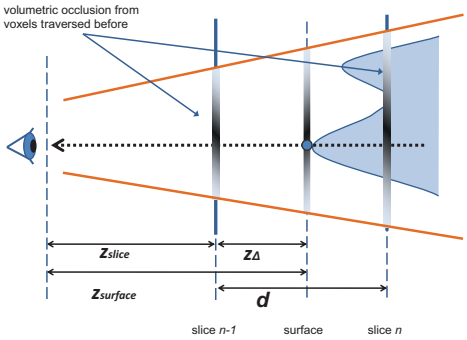

Figure 4: Illustration of the basic geometric setup for updating the partial occlusion buffer of a geometric surface at a view space distance $z_{surface}$ which is situated between slices $slice_{n-1}$ (at view space distance z_{slice}) and $slice_n$, separated by the slice distance d. The partial occlusion of the surfaces is approximated by interpolating between $volumetric\_occlusion_{n-1}$ and $volumetric\_occlusion_n$ based on z_Δ.

G-Buffer, as discussed in Section 3.1.1, is used to increase performance by restricting updating of the partial occlusion buffer to pixels containing actual surfaces. The partial occlusion buffer is repeatedly overwritten until fragments of a slice behind the surface are discarded by the depth test, at which point the stencil buffer will be cleared for that fragment, preventing further updates. The computation of the partial occlusion buffer completes after traversing last slice and as such then contains the occlusion shadows cast from the *volume* onto the *geometry*.

3.1.5 Computing the Vicinity Occlusion Term

Algorithm 5 VicinityOcclusion($geom\_linear\_depth$)

```
occlusion ⇐ 0
z_surface ⇐ Texture2D(geom_linear_depth, f_t)
biased_depth ⇐ z_surface + depth_bias
for all samples p_i within the vicinity occlusion extent R_t do
    project p_i into texture space t_i using z_surface
    depth_sample ⇐ Texture2D(geom_linear_depth, t_i)
    if depth_sample > 0 ∧ depth_sample < biased_depth then
        occlusion ⇐ occlusion + 1
    end if
end for
return 1 − occlusion / |t_i|
```

The vicinity occlusion term is determined by rendering a full screen quad while comparing the depth of each pixel with the depths of neighboring pixels in the geometric depth buffer, counting neighboring pixels with greater depth as occluded, as outlined in Algorithm 5. The ratio of occluded to non-occluded pixels is an approximate geometric occlusion term for the high frequency geometric details. The depth value of the current pixel is offset to avoid self occlusion, similar in spirit to the depth bias used to prevent equivalent artifacts using shadow maps to compute shadows of solid geometry [22].

The directional occlusion shading model only considers the occlusion from structures within a cone with opening angle θ, as illustrated in Figure 5. An occluder with view space distance l from the surface must be within the base of the cone of radius $r = l \cdot \tan(\theta)$ in order to impact the surface at view space distance $z_{surface}$. The view space radius $r_{volumetric\_occlusion} = d \cdot \tan(\theta)$ used for blurring the previous volumetric occlusion buffer, is thus determined by the cone opening angle θ and the inter-slice distance d.

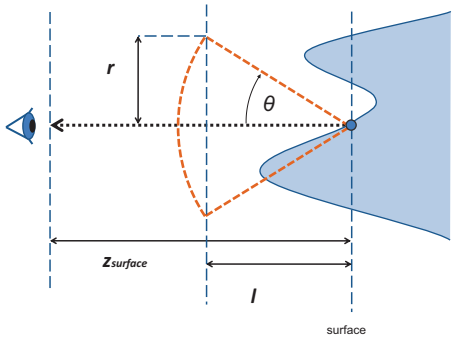

Figure 5: Illustration of the basic geometric setup for computing the vicinity occlusion term of a geometric surface at the view space distance $z_{surface}$.

Computing the vicinity occlusion term, in contrast to computing the volumetric occlusion term, however does not intrinsically lead to values for l (and indirectly to the radius of influence r). Other screen space ambient occlusion techniques typically let the user specify r directly in order to change the size of the occlusion effects. This however would decouple the qualitative appearance of the vicinity occlusion effects from the volumetric occlusion effects. Instead, a "virtual" slice distance $l_{vicinity}$ is specified by the user and utilized to compute the vicinity occlusion extent $r_{vicinity\_occlusion} = l_{vicinity} \cdot \tan(\theta)$. A "virtual"slice distance $l_{vicinity} = 0.005$ was used in Figures 6(b), 6(d) and 6(f) to show occlusion effects both on the surface and the volume which change depending on the values of the blur angle $\theta \in \{25°, 45°, 65°\}$.

The view space vicinity occlusion extent $r_{vicinity\_occlusion}$ for a surface at view space distance $z_{surface}$ is then projected into image space by the (perspective) projection matrix. Similar to the volumetric occlusion shading method, a number of samples are then generated within the projected disk and used to sample the depth buffer in order to compute the vicinity occlusion term.

Incrementally filtering the volumetric occlusion buffers allows minimization of the number of samples since each individual filter step only covers a rather small number of texels of the previous volumetric occlusion buffer. In contrast, computing the vicinity occlusion requires the consideration of a much larger area in the buffer containing the view space distances of the geometry. Using a Poisson distribution [4] with up to 118 samples proved to sufficiently capture the high frequency surface-to-surface occlusion effects and were thus used generating the results presented in Section 4. The vicinity occlusion term is capturing the occlusion effects received by the *geometry* from the surrounding *geometry*.

Other, more general screen space ambient occlusion techniques geared towards high performance shading of screen covering geometric scenes often employ complex approaches in order to achieve high visual quality at high frame rates. The vicinity occlusion term however is only computed for pixels that contain geometric structures; pixels not covered by the geometric structures are shaded by the direct volume rendering term, which is the major performance limiting part of the presented algorithm. This, combined with the sparsity of the DTI fiber tracts and streamlines reduces the contribution of the vicinity occlusion term to the overall performance and allows an easy to implement vicinity occlusion term that also emphasizes the high frequency details of the geometric structures.

This approach works especially well for the high frequency tube models, since those are typically bundled close together, where the vicinity occlusion term is capturing their mutual occlusion effects with sufficient fidelity. This is not always the case when considering general geometric models, which often cover large spans of the

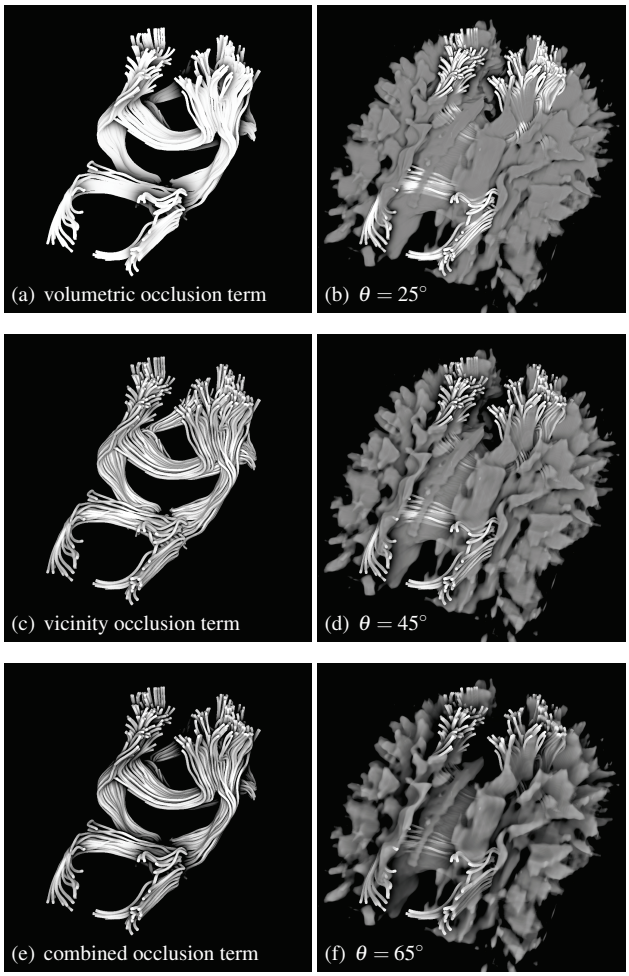

Figure 6: Left column: DTI fiber tracts were rendered where the transfer function was set to render all voxels as transparent in order to emphasize the occlusion effects on the geometric structures. a) shows the volumetric occlusion term, c) shows the vicinity occlusion term and e) shows the combined occlusion term. The right column shows the combined occlusion term for various values of θ.

domain. The vicinity occlusion term is not adequately considering those, due to its local nature; more general screen space ambient occlusion approaches should be used to for those types of geometry. Changing the blur angle of the volumetric occlusion term can also capture the occlusion effects of remote structures to a certain extent, namely those that are within the blur cone.

3.1.6 Final Compositing and Shading

The last step of our proposed shading method is to combine the partial occlusion term with the vicinity occlusion term in order to yield a final combined occlusion term for shading the geometric surfaces. Figure 6(a) shows the volumetric occlusion term that captures only the shadows of *low frequency* geometric details plausibly, which is in contrast to the vicinity occlusion term (Figure 6(c)) that approximates occlusion effects for *high frequency* local geometric structures. The combination of the volumetric occlusion term containing occlusion effects from the volume and the low frequency geometric structures with the vicinity occlusion term then provided for both local, *high frequency* and global *low frequency* occlusion effects, as demonstrated in Figure 6(e). We found that multiplying the volumetric and the vicinity occlusion terms allows the vicinity

occlusion terms of partially occluding voxels to continue providing depth cues on the surface, which tend getting lost using the minimum. The G-Buffer is then used to compute the ambient, diffuse and specular terms of the surface lighting, which are then blended together with the combined occlusion term and the surface color and the shaded volume, as detailed in Algorithm 6.

Algorithm 6 CompositeBuffers

```
SetRenderTarget(window_frame_buffer)
for all fragments f of full screen quad do
  volume_color ⇐ Texture2D(eye_bufferf_clip)
  position ⇐ Texture2D(geom_position,f_clip)
  normal ⇐ Texture2D(geom_normal,f_clip)
  surface_color ⇐ Texture2D(geom_color,f_clip)
  partial_occlusion ⇐ Texture2D(partial_occl_buffer,f_clip)
  if f covers geometry then
    vicinity_occlusion ⇐ VicinityOcclusion(geom_linear_depth)
    {Algorithm 5}
    occlusion ⇐ vicinity_occlusion · partial_occlusion
    (amb, diff, spec) ⇐ Phong(position , normal )
    shaded_surface ⇐ occlusion · surface_color · (amb + diff) +
    spec
    frame_buffer ⇐ (shaded_surface · (1 − volume_color_a) +
    volume_color_rgb), 1)
  else
    frame_buffer ⇐ volume_color
  end if
end for
```

4 Results and Discussion

The method was implemented using OpenGL and Cg running on an NVIDIA GeForce GTX 580 GPU with 1.5 GB of video memory. The images were rendered into 16-bit precision floating-point buffers. The tube geometry was computed by sweeping a circle along a B-spline curve approximating the individual line segments. The surface positions, colors and normals were evaluated on the CPU along the curve and then stored in video memory.

Figure 1 shows a chemical simulation data set where two chemical reagents are mixed in order to accelerate their reaction. The streamlines were color mapped with the mixture fraction denoting the relative ratio of two chemical agents shown in red and blue. A 2D transfer function was used to highlight the boundaries of the mixing pipe colored white, while at the same time showing the concentration of the product as the result of the chemical reaction. Figure 1(a) shows the combined data set using volumetric Phong shading with on-the-fly gradient estimation together with streamlines rendered with Phong surface shading which make it difficult to gain insight into the complex turbulent behavior of the streamlines. Figure 1(b) combines the vicinity occlusion term with the volumetric occlusion of the scalar field to obtain mutual occlusion effects which provide additional context between the mixture fraction, visualized on the streamlines, and the concentration of the chemical product, which is visualized by the volume rendered scalar field.

Figure 7 shows an MRI data set with registered DTI fiber tracts rendered without occlusion effects in Figure 7(a), which makes it hard to gain insight about the relative spatial arrangement between the geometric structures and the volumetric features. Figure 7(b) uses the combined occlusion shading to render physically plausible occlusion effects which impact both the volume and the geometry, thus enhancing the depth perception and visual comprehension of the data set. Being able to precisely reason about the arrangement of structures (neural fibers, sensitive tissue) within the brain is of key importance to surgical planning, where surgical instruments must be placed with uttermost precision, since minimal deviations could damage sensitive tissues, especially when considering brain surgery.

Figure 8 demonstrates a single time step of an astrophysical data set showing the magnetic field of a Sun-like star undergoing an inversion of its magnetic poles [2]. The magnetic field lines were color mapped with the polarity of the longitudinal magnetic field with red and blue showing positive and negative polarity respectively. The scalar volume shows the magnitude of the magnetic field mapped to red, orange and white. Figure 8(a) shows the combined data set rendered with volumetric Phong shading, which makes it difficult to gain insight into the complex configuration of the magnetic field lines with respect to the magnetic field strength. Volumetric Phong shading is also dependent on the volume gradient for its shading, which is problematic for homogeneous or noisy regions since there, the gradients are either not defined, or point into incoherent directions, as observable in the top right part of Figure 8(a). The combined rendering with occlusion effects, as shown in Figure 8(b), visualizes the volume showing the magnetic field strength together with the magnetic field lines illustrating the polarity of the magnetic field on the field lines themselves. The occlusion effects allow better comprehension of the structure of the field lines, while at the same time benefiting from the context providing scalar volume. Its independence from the volumetric gradient avoids the artifacts of using volumetric Phong shading.

Table 1 shows the performance behavior of the combined geometric and volumetric occlusion shading method compared to volume shading with an emission-absorption or Phong shading model with on-the fly gradient estimation. The performance of the combined shading method is reduced compared to the traditional volume shading methods, which is similar to the performance behavior of the original purely volumetric occlusion shading method. Introducing the geometric structures into the occlusion shading computation reduces the performance by about 25% compared to computing volumetric occlusion shading alone, by about 86% compared to computing emission/absorption and by 71% compared to computing combined Phong shading. Despite this, sufficiently high performance is maintained and as such allows interactive exploration of the data sets. The computation of the vicinity occlusion term alone performs at interactive frame rates, despite the high number of samples taken and does not constitute the performance limiting aspect of the presented combined shading method, which instead is the incremental filtering inherent to the directional occlusion shading method. The image space approach for computing the vicinity occlusion term is especially beneficial for the astrophysical data set with its 7.9 million triangles which renders only at half of the speed, compared to the DTI data set which has a fifth of the number of triangles.

5 Conclusion and Future Work

An extension to the volumetric directional occlusion shading method was been presented which renders occlusion effects of geometric structures combined with those of volumetric origin. Both geometric and volumetric structures act as shadow casters and shadow receivers and their shadows are computed by modifying the occlusion buffer update of the original directional occlusion shading method [14]. An additional surface based vicinity occlusion term is computed to capture high frequency shadowing effects between fine detailed geometric structures, such as streamlines or DTI fibers, thus supplementing the volumetric occlusion term which captures the low frequency occlusion effects of the geometry and volume contained in the scene. The extended method, similar to the volumetric directional occlusion shading algorithm, does not rely on pre-computation and thus enables interactive manipulation of camera position, transfer function and the geometry as well, which are all taken into account during the occlusion computation. The required extensions reduce the rendering performance by about 25% compared to only computing the occlusion shading of the volumetric structures and as such allow interactive exploration of geometric and volumetric data

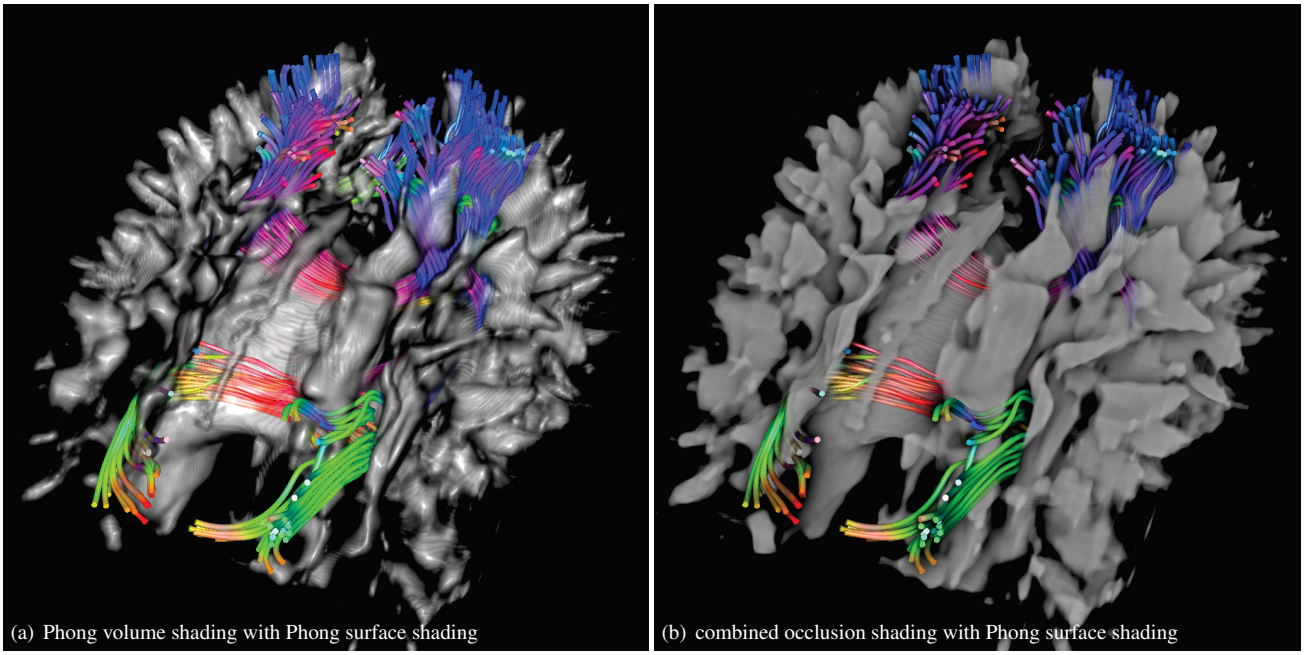

Figure 7: DTI fiber tracts (720k triangles) and a registered MRI volume ($96 \times 96 \times 81$ resolution) rendered with a) Phong volume shading with on-the-fly gradient estimation and Phong surface shading and b) combined directional occlusion shading and Phong surface shading.

Table 1: Performance for various rendering methods and data sets, measured in FPS on an NVIDIA GeForce GTX 580 GPU.

data set	mixing pipe (Figure 1)		DTI fibers (Figure 7)		astrophysical (Figure 8)	
slices, resolution, triangles	574, $1800 \times 121 \times 121$, 137k		512, $96 \times 96 \times 81$, 720k		961, $256 \times 512 \times 512$, 7.9M	
θ, volumetric samples, vicinity samples	45°, 2×2, 118		45°, 4×4, 81		65°, 2×2, 81	
image resolution	768×128	1153×256	512×512	1024×1024	512×512	1024×1024
vicinity occlusion shading only	102.87	102.35	106.66	103.78	67.05	54.41
volumetric occlusion shading only	16.75	16.67	11.24	3.49	10.61	5.78
combined occlusion shading	11.73	11.79	9.73	3.09	7.31	4.54
emission/absorption & Phong	101.26	101.1	82.61	26.9	36.31	22.75
volumetric Phong & Phong	101.93	51.76	32.22	8.82	18.41	9.43

sets while increasing the depth perception and scene comprehension.

In the future, we would like to allow the user to change the light position with greater flexibility, in contrast to the light source being fixed at the viewer. The multi-directional occlusion shading method by Šoltészová *et al.* [20] could be incorporated into the combined rendering method by changing the filtering of the volumetric occlusion buffer appropriately. Similarly, weighting of the samples could be done for the vicinity occlusion term as well in order to qualitatively match the shadowing of the volume and the geometry.

Acknowledgements

This research was sponsored by the National Nuclear Security Administration under the Advanced Simulation and Computing program through DOE Cooperative Agreement #DE-NA0000740, and by Award No.KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST), and DOE SciDAC:VACET, NSF OCI-0906379. We would also like to acknowledge NVIDIA for hardware donations and the reviewers for their thoughtful comments.

References

[1] L. Bavoil, M. Sainz, and R. Dimitrov. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 talks*, SIGGRAPH '08, pages 22:1–22:1, New York, NY, USA, 2008. ACM.

[2] B. P. Brown, M. S. Miesch, A. S. Browning, M. K. andBrun, and J. Toomre. Magnetic Cycles in a Convective Dynamo Simulation of a Young Solar-type Star. *The Astrophysical Journal*, 731:69–+, Apr. 2011.

[3] J. Díaz, H. Yela, and P. Vázquez. Vicinity occlusion maps: Enhanced depth perception of volumetric models. In *Computer Graphics International*, 2008.

[4] D. Dunbar and G. Humphreys. A spatial data structure for fast poisson-disk sample generation. *ACM Trans. Graph.*, 25(3):503–508, 2006.

[5] M. H. Everts, H. Bekker, J. B. T. M. Roerdink, and T. Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1299–1306, 2009.

[6] V. Interrante and C. Grosch. Strategies for effectively visualizing 3d flow with volume lic. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 421–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[7] 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.

[8] M. McGuire, B. Osman, M. Bukowski, and P. Hennessy. The alchemy screen-space ambient obscurance algorithm. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 25–32, New York, NY, USA, 2011. ACM.

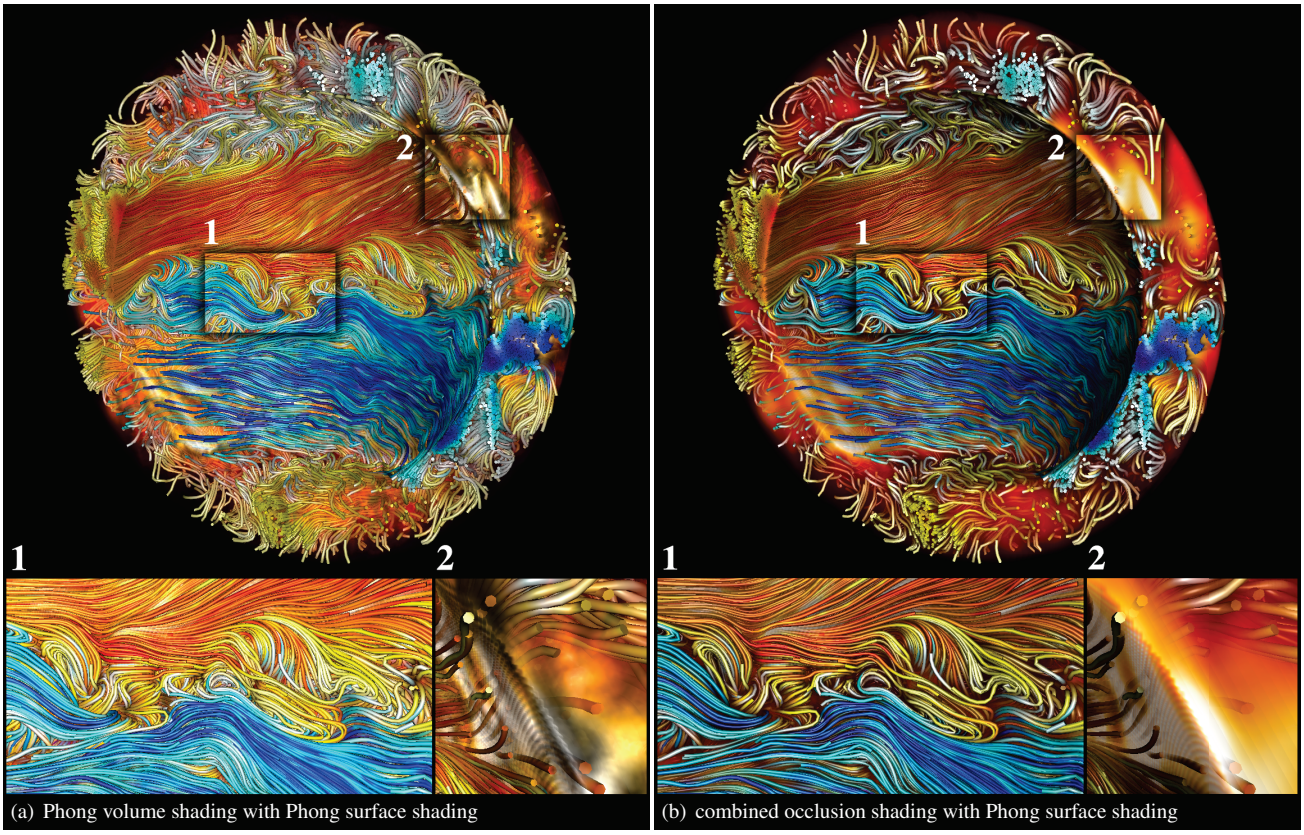

Figure 8: An astrophysical data set was used to compute magnetic field lines (7.9M triangles) with Phong surface shading and a scalar volume ($256 \times 512 \times 512$ resolution) showing the magnitude of the magnetic field, rendered with a) Phong volume shading with on-the-fly gradient estimation and Phong surface shading, and b) combined directional occlusion shading and Phong surface shading.

[9] Z. Melek, D. Mayerich, C. Yuksel, and J. Keyser. Visualization of fibrous and thread-like data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1165–1172, 2006.

[10] M. Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 97–121, New York, NY, USA, 2007. ACM.

[11] D. Patel, S. Bruckner, I. Viola, and E. Gröller. Seismic volume visualization for horizon extraction. In *PacificVis*, pages 73–80, 2010.

[12] T. Saito and T. Takahashi. Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '90, pages 197–206, New York, NY, USA, 1990. ACM.

[13] M. Schott, A. Pascal Grosset, T. Martin, V. Pegoraro, S. T. Smith, and C. D. Hansen. Depth of field effects for interactive direct volume rendering. In *Computer Graphics Forum (Proceedings of Eurographics/IEEE VGTC Symposium on Visualization 2011)*, volume 30, pages 941–950, 2011.

[14] M. Schott, V. Pegoraro, C. Hansen, K. Boulanger, and K. Bouatouch. A directional occlusion shading model for interactive direct volume rendering. In *Computer Graphics Forum (Proceedings of Eurographics/IEEE VGTC Symposium on Visualization 2009)*, volume 28, pages 855–862, 2009.

[15] G. Schussman and K.-L. Ma. Scalable self-orienting surfaces: A compact, texture-enhanced representation for interactive visualization of 3d vector fields. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 356, Washington, DC, USA, 2002. IEEE Computer Society.

[16] P. Shanmugam and O. Arikan. Hardware accelerated ambient occlusion techniques on gpus. In *In I3D Š07: Proceedings of the 2007 symposium on Interactive 3D graphics and games, ACM*. Press, 2007.

[17] C. Stoll, S. Gumhold, and H.-P. Seidel. Visualization with stylized line primitives. *Visualization Conference, IEEE*, 0:88, 2005.

[18] A. Stompel, E. B. Lum, and K.-L. Ma. Visualization of multidimensional, multivariate volume data using hardware-accelerated non-photorealistic rendering techniques. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 394, Washington, DC, USA, 2002. IEEE Computer Society.

[19] F. V. Three-Dimensional, G. shi Li, U. D. Bordoloi, and H. wei Shen. Chameleon: An interactive texture-based rendering framework. In *in Proceedings IEEE Visualization 2003. IEEE, 2003*, pages 241–248, 2003.

[20] V. Šoltészová, D. Patel, S. Bruckner, and I. Viola. A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum*, 29(3):883–891, J 2010.

[21] A. Wenger, D. F. Keefe, S. Zhang, and D. H. Laidlaw. Interactive volume rendering of thin thread structures within multivalued scientific datasets. *IEEE Transactions on Visualization and Computer Graphics*, 10:2004, 2003.

[22] L. Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274, 1978.

[23] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive visualization of 3d-vector fields using illuminated stream lines. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 107–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.