LineAO—Improved Three-Dimensional Line Rendering

Sebastian Eichelbaum, *Member*, *IEEE Computer Society*,
Mario Hlawitschka, *Member*, *IEEE Computer Society*, and Gerik Scheuermann, *Member*, *IEEE*

Abstract—Rendering large numbers of dense line bundles in three dimensions is a common need for many visualization techniques, including streamlines and fiber tractography. Unfortunately, depiction of spatial relations inside these line bundles is often difficult but critical for understanding the represented structures. Many approaches evolved for solving this problem by providing special illumination models or tube-like renderings. Although these methods improve spatial perception of individual lines or related sets of lines, they do not solve the problem for complex spatial relations between dense bundles of lines. In this paper, we present a novel approach that improves spatial and structural perception of line renderings by providing a novel ambient occlusion approach suited for line rendering in real time.

Index Terms—GPU, ambient occlusion, illumination, diffusion tensor data, fiber tracking, streamline visualization

1 Introduction

In many areas of visualization, line rendering techniques play an important role. In flow visualization, 3D vector fields are visualized using streamlines, streaklines, or pathlines. They have a direct physical meaning and are used to understand simulated and measured data in many parts of engineering. Lines are used to represent electric and magnetic fields as well as velocity fields in a very intuitive way. Even in tensor fields, where a line tangent is not given explicitly, tensor lines and hyperstreamlines [1], [2] are used to display important structures of the field.

But vector and tensor fields do not only play a crucial role in flow visualization. In medical visualization, a large variety of measuring and imaging methods, such as electroencephalograhy (EEG) or magnetic resonance tomography (MRT), are available. From EEG, it is possible to derive the describing electric field in the head. Using diffusion tensor imaging (DTI) or high angular-resolution diffusion imaging (HARDI), it is possible to coarsely reconstruct the neuronal connections inside the brain and to obtain a better understanding of its structure. These techniques are called fiber tracking or tractography [3], [4], [5], [6], [7] and are often based on enhanced streamline techniques.

Each kind of line data, each visualization technique, and each scientific use case has its own specific properties and constraints. Mostly, large and dense line data are given and the structural relation of bundles of lines as well as local shape information is crucial for understanding the represented structures. For exploring these kind of data, filtering and rendering of line data in real time are an important requirement for modern visualization techniques. Besides this, consistency of rendered images under modification and interaction with the data is important to retain the mental image of its structure.

Standard line rendering techniques often use local illumination and shading to emphasize shape or simple spatial relations on lines but have severe limitations toward the simultaneous display of local and global structural detail and spatial relations. Although ray-tracing techniques are able to provide realistic lighting models, they do not serve the real time and interactivity requirement.

With LineAO, we contribute a novel approach, which overcomes these problems and provides a greatly *improved structural and spatial perception* for the rendered line data in a very intuitive and natural way, as demonstrated in Fig. 1. It uses ideas from ambient occlusion (AO) and global illumination to allow a *simultaneous depiction of local and global line structures*. It is known that global lighting effects are very important for determining an object's position and spatial relations [9], [10], [11], [12]. Since real-time ability and dynamic scenes are a major demand in many fields, our method *renders in real time, without precomputation* and is, therefore, capable of being used in explorative tools, where the researcher interactively modifies the line data. We combine global ambient lighting with the scattered light contributions from surrounding lines and adhere to the intrinsic fixed line width on screen, ensuring *consistency under modification and interaction*.

LineAO is not appropriate for 2D line data with no or little overlapping or de facto 2D line renderings. Additionally, LineAO is not a physical lighting model. It is an approach, which uses global illumination effects to allow scientists to perceive structural and spatial properties of scientific line data in a natural way.

- *S. Eichelbaum and G. Scheuermann are with the Abteilung für Bild- und Signalverarbeitung, Universität Leipzig, PF 100920, D-04009 Leipzig, Germany. E-mail: {eichelbaum, scheuermann}@informatik.uni-leipzig.de.*
- *M. Hlawitschka is with the Lehrstuhl für Wissenschaftliche Visualisierung, Universität Leipzig, PF 100920, D-04009 Leipzig, Germany. E-mail: hlawitschka@informatik.uni-leipzig.de.*

Manuscript received 27 Aug. 2011; revised 27 Jan. 2012; accepted 30 May 2012; published online 8 June 2012.
Recommended for acceptance by G. Drettakis.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-08-0202.
Digital Object Identifier no. 10.1109/TVCG.2012.142.

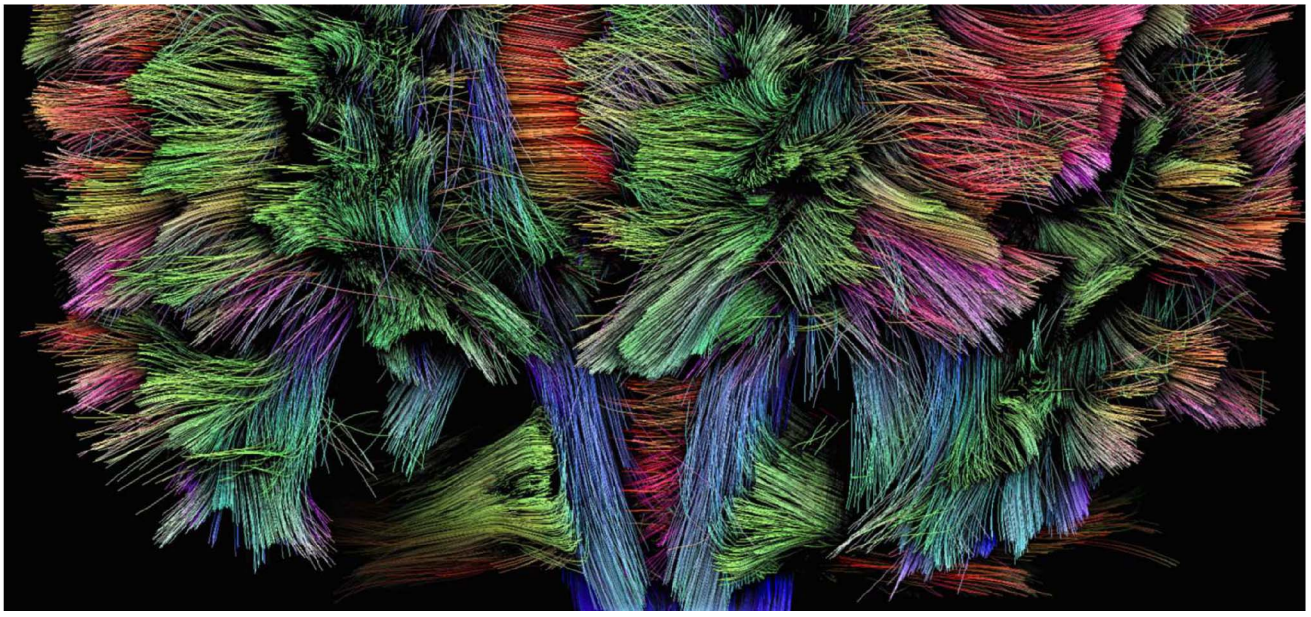


Fig. 1. Fiber tractography rendered using LineAO and colored according to the local tangent direction [8], which is very common in neuroscience. The improved perception of spatial relations between and in bundles of lines can be seen especially well in the brain stem (center bottom part of the image), where the *Pons* and *Medulla Oblongata* pass into the *Spinal Cord*. The layering of these bundles as well as the fissure structure in the *Frontal Lobe* can be observed very distinctly.

2 Background

2.1 Line Rendering: State of the Art

Current line data rendering approaches usually employ local illumination models for lines to emphasize the shape of lines and line bundles [13], [14]. These techniques apply a simplified Blinn-Phong [15] shading, which provides diffuse reflection and specular highlights and allows depiction of local shape features. An alternative approach is to create the look of real cylindrical tubes by rendering line data using quad strips [16], [17], or triangle strips [18]. These approaches allow correct lighting (according to Phong's lighting model) and keep the density of the rendered lines while zooming, which often is a requirement.

Besides shading techniques, there are approaches which utilize depth cueing and haloing to provide further structural information. In [19], depth-dependent halos are rendered around lines to emphasize tight bundles of lines, and the additional depth cueing further increases depth perception in this method. Unfortunately, due to the heavy overlapping of halos, this type of depth cueing loses its effect if the line data are very dense.

Another approach is to interpret dense line data as volumetric data. Schussman and Ma [20] propose a method for sampling extremely dense line data and rendering them with direct volume rendering. Unfortunately, this method does not focus on spatial perception in the final volume rendering.

Hair rendering is another shading technique for dense line data, which relies on simplifications implied by their underlying model: All hair rendering techniques are optimized to mimic the effects of light scattering in a multitude of thin, translucent hair without caring about the exact shading of each single strand of hair. Therefore, most of the geometric simplifications that make hair rendering efficient cannot be used in our approach since, in scientific data, each single line's shading is important. For a comprehensive overview and up-to-date shading techniques, we refer to [21], [22], [23], [24], [25].

Although most current approaches are able to depict local shape of line data, they are not able to properly represent spatial relations and the local structure in bundles of dense line data at the same time.

2.2 Global Illumination in Visualization

Global illumination techniques and ambient occlusion have found their way into more and more scientific visualization techniques. For example, in direct volume rendering, global illumination and occlusion-based shading can help to provide the required cues to keep track of spatial relations in the rendered images. Ruiz et al. [26] used the idea of obscurance to provide realistic volume renderings. Other methods can provide dynamic illumination for direct volume rendering [27], [28]. Wyman et al. applied global illumination techniques to interactive isosurface rendering using precomputed radiance transfer [29]. In the work of Melek et al. [30], ambient skylight and shadows are used to emphasize structure in fibrous microscopy images.

Especially in medical applications, these techniques help to understand structure and relations in 3D anatomical data. Besides the medical use case, chemical and biochemical visualization also profits from more realistic rendering and, e.g., [31] and [32] added ambient occlusion for a better structural perception of very complex molecule structures. Gribble and Parker applied ambient occlusion to particle rendering and investigated its effect in a formal user study [33].

2.3 Ambient Occlusion

Roughly spoken, Ambient Occlusion represents the diffuse lighting effect on a day with overcast sky. Proper ambient light is a very complex and globally defined problem, since

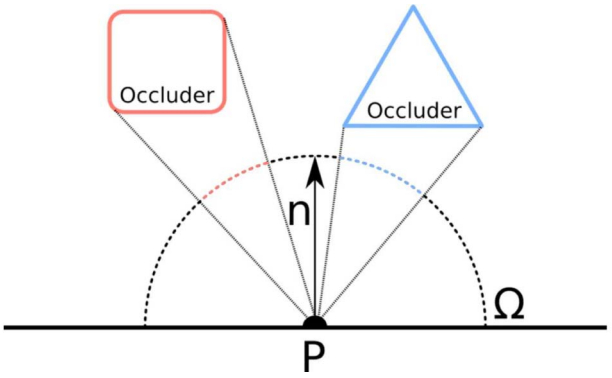


Fig. 2. Illustration of (1). Each geometry in the scene can occlude a part (red and blue) of the unit hemisphere around P. This fraction weighted by the relative direction to the surface normal describes the ambient occlusion of the small surface element represented by the normal n at point P.

the whole scene defines the distribution of ambient light. For this reason, real-time computer graphics used only direct illumination for a long time, where the ambient light is assumed to be constant at every point in the scene. AO estimates the ambient light distribution in a complex scene to imitate radiance of light on nonreflective surfaces. This increases the realism of computer generated images and improves the perception of relations between objects [9], [10], [11], [12].

In this section, we give a short introduction to the fundamentals of AO and an overview on current AO techniques. We will explain why these techniques are not useful for line rendering in real time so far.

2.3.1 Model of Ambient Occlusion

The AO factor describes the amount of ambient background light not reaching the surface. In other words, it determines, how much of a surface is concealed by other surfaces, prohibiting ambient light to reach the surface. To describe this mathematically, we locally define a surface using its tangential plane at point P with surface normal n. To measure the amount of occlusion for the point P, it is required to measure the surface area of a unit hemisphere occluded by surrounding objects as demonstrated in Fig. 2. Mathematically, this surface area is defined by the integral over the unit hemisphere. The actual check whether a point on the hemisphere is occluded or not is done by a binary visibility function $V(\omega, P)$, being 1 if the surface point is visible, and 0 if it is occluded. The unit vector ω hereby samples the unit hemisphere by pointing from P to the surface point of the hemisphere Ω.

The amount of light energy reaching a point on the surface is defined by the angle between the light direction and the surface's normal. Using this, AO defines the amount of skylight energy *not* reaching P due to the occlusion. This yields the standard ambient occlusion definition used in literature [34], [35], [36], [37]

$$AO(P,n) = \frac{1}{\pi}\int_{\Omega}(1 - V(\omega, P))\langle\omega, n\rangle d\omega, \qquad (1)$$

for point P on a surface and its normal n. Due to the influence of the scene in the visibility function, it is obvious that there is no easy analytical solution to the integral. Therefore, it is usually approximated and, for reasonable setups of the geometry, can be approximated by the Riemann sum

$$AO(P,n) = \frac{1}{\pi}\lim_{s\to\infty}\sum_{i=1}^{s}(1 - V(\omega_i, P))\langle\omega_i, n\rangle\frac{\pi}{s}, \qquad (2)$$

and truncated to a series of s samples

$$AO(P,n) \approx AO_s(P,n) = \frac{1}{s}\sum_{i=1}^{s}(1 - V(\omega_i, P))\langle\omega_i, n\rangle, \qquad (3)$$

which approximates (1) for a sufficiently high sample count s and a well-chosen distributions of $\omega_i \in \Omega$ on the unit hemisphere.

2.3.2 Current Ambient Occlusion Techniques

To solve the above equation for complex scenes, many approximative algorithms have evolved. Their main goal is to efficiently evaluate the visibility function V for complex and large scenes. Thereby, these methods can be classified in ray-tracing approaches, which try to approximate the AO effect on a physical basis and real-time approaches which try to achieve the AO effect phenomenologically. In this section, we give an overview on these methods and their drawbacks regarding line rendering.

Ray-tracing approaches often use proxy geometry to reflect the rather complex scene with simpler primitives. This way, occluders can be described and tested faster by the visibility function V. These proxy primitives are often analytic objects such as spheres or discs [34], [38], [39], [40], [41], [42] or more complex primitives that better match the given geometry [43], [44]. These physically based approaches often include heavy precomputation steps and produce an overestimation of the AO effect due to the approximative scene description. Due to the precalculation step, these methods are mostly limited to static scenes and only allow limited interaction and interactivity, which is why we do not further consider them.

Unlike physically correct techniques, the class of phenomenological approaches is able to run in real time. The cornerstone of most of these techniques is the image enhancement using unsharp masking of the depth buffer, which has been presented by Luft et al. [45]. In their method, the depth buffer of the rendered scene is interpreted as height field and compared with a low-pass-filtered copy. This is utilized to provide information about spatially important areas, which then allows a modification of several image properties such as local contrast. Although this is no real ambient occlusion effect, it provides a remarkable improvement in spatial perception and spawned a whole set of methods grouped under the term screen space ambient occlusion (SSAO). A widely known SSAO technique is CryTec SSAO [46], [47]. It adapts unsharp masking and uses sparse sampling of the depth buffer to derive visibility information in the neighborhood of a point in the scene, based on information available in screen space. It uses the depth information in the neighborhood of a pixel to estimate the amount of ambient light that reaches the corresponding point on the surface. Due to the limited resolution in screen space, this approach samples the ambient occlusion factor at

a low angular resolution, leading to inaccurate, results especially for small or distant objects. It can be extended by using better sampling schemes [48], [49] or by adding better filtering and distant occluders [35]. Two of the main disadvantages of these methods are the low spatial resolution and the noise induced due to the stochastic sampling scheme. Hoang and Low [50] introduced a multiresolution approach to combine the AO effects of distant objects with detailed local AO.

To circumvent the spatial resolution problem, hybrid approaches have been developed that combine ray tracing a simplified scene with SSAO [36], [51]. Other methods precompute additional fields or acceleration structures abstracting the occlusion caused by objects [52], [53], [54] or precalculate the transfer of incident light from an environment map into the incident radiance on the surfaces [55], [56], [57]. Unfortunately, these methods suffer in being constrained to static scenes or require precomputation steps.

Each of the above-mentioned techniques has its limitations toward some of the render properties we mentioned in Section 1. *Physically based approaches* fail to provide real-time rendering or rely on heavy precomputation steps. As these techniques aim at physically correct shading, they cannot emphasize structure in line data using a physically incorrect but detail-emphasizing, illustrative shading.

Phenomenological approaches work well in compact scenes with objects with a certain minimal volume, due to the fixed sampling radii and low spatial resolution. The low spatial resolution of these approaches prohibits the proper shading of small and thin structures in line data due to aliasing. Although the multiresolution SSAO approach [50] can solve the problem of low spatial resolution by sampling at multiple scales, it reaches its limits when applied to thin structures [58]. It does not modify the AO approach itself and thus, does not incorporate any special obscurance weight that allows for emphasizing global structures while retaining the local detail in these line bundles. Additionally, phenomenological approaches often are not able to create coherent renderings when zooming, as line bundles get less dense when zooming. This is due to the intrinsic fixed line width on the screen, which stays constant during zoom.

3 Method

In the previous section, we have summarized the state of the art in line rendering, introduced the mathematical fundamentals behind AO, and given an overview on available approaches for ambient occlusion rendering.

In this section, we introduce LineAO and show how LineAO uses the intrinsic fixed line width on screen for visibility evaluation while ensuring consistency under zoom and interaction. We introduce a sampling scheme to solve the problem of low spatial resolution in screen space and provide an obscurance weight tailored toward line rendering to furthermore emphasize local detail in bundles while retaining global structural shading. It prohibits heavy darkening of local structures due to large, global structures. We finally provide a pragmatic implementation guideline and show how we combine LineAO with illuminated lines and tube-based rendering.

3.1 LineAO Sampling Scheme

As we aim at a screen-space solution for visibility evaluation, we need to handle the problem of low spatial resolution, common to nearly all SSAO approaches. To achieve this, sampling on a single hemisphere is not sufficient. We need to sample near and distant lines and classify them into levels of distance. We, therefore, begin to extend the sampling scheme, weighting, and view evaluation in (3) to increase spatial resolution, while tailoring it toward correct handling of local and distant occluders. First, we replace the orientation-based weighting, with a custom obscurance weight g, which attenuates the occlusion term $1 - V$:

$$AO_s(P) = \frac{1}{s}\sum_{i=1}^{s}[(1 - V(\omega_i, P))g(\omega_i, P)]. \tag{4}$$

To additionally allow evaluation of AO on multiple hemispheres, we include a parameter r, defining the radius of the hemisphere used to sample the surrounding objects. We call

$$AO_s(P, r) = \frac{1}{s}\sum_{i=1}^{s}[(1 - V(r\omega_i, P))g(r\omega_i, P)] \tag{5}$$

the local AO for a hemisphere with the radius r. Although the weighting function g and the hemisphere radius r now allow the combination of AO for multiple hemispheres, the radius is not sufficient for classifying distance levels and, therefore, the different effects of local and global occluders to the current point P. To accommodate this classification issue, we modify the visibility and weighting function to depend on a parameter l, defining the level of distance. The smaller l, the more local detail is emphasized. The larger, the more global structures are important. It allows us to handle local and distant occluders properly using V and g. This defines an ambient occlusion term for a given distance level l, which accommodates for the different properties of distant and near occluders:

$$AO_{s,l}(P, r) = \frac{1}{s}\sum_{i=1}^{s}[(1 - V_l(r\omega_i, P))g_l(r\omega_i, P)]. \tag{6}$$

This yields the final evaluation of the AO fraction of one hemisphere with the radius r for a point P with s_h samples and distance level l. In contrast to other approaches, we are able to handle distant occluders and local structure directly with our specialized obscurance weighting function g, without the use of proxy geometry as, e.g., in [35].

With (6), we are now able to classify each occluder into distance levels and to combine their AO effect using

$$\mathrm{LineAO}_{s_r, s_h, r_0}(P) = \sum_{j=0}^{s_r - 1} AO_{\frac{s_h}{j+1}, j}(P, r_0 + jz(P)). \tag{7}$$

LineAO evaluates (6) s_r times for increasing sampling hemisphere radii and distance level l. The first iteration $j = 0$ represents the smallest hemisphere, which is responsible for the local details, and uses s_h samples on the hemisphere. For the following iterations, the term $\frac{s_h}{j+1}$ ensures that the number of samples per hemisphere is reduced. As LineAO sums up the occlusion effects of near and distant occluders and ignores the possibly overlapping occlusions on different hemispheres, it generates the intended effect of heavier

occlusion in very dense areas. The term $r_0 + jz(P)$ hereby defines the increasing hemisphere radius for increasing distance levels. Since j is zero for the first level, r_0 defines the smallest hemisphere for sampling in the direct vicinity of a line. The zoom function $z(P)$ represents the relation between the supposed line volume in world space and the *intrinsic fixed line width*on screen and, therefore, creates a coherent AO effect when zooming. Finally, LineAO is clamped to the interval $[0, 1]$.

In this section, we have presented our sampling scheme, which handles the problem of low spatial resolution. With this, we are able to combine the advantages of local detail with perception of global structure in line data, due to an adaptive spatial resolution, depending on the distance level. To achieve the goals mentioned in Section 1, namely emphasizing local detail and global structure of line data simultaneously, we need to define a weighting function g tailored toward this. In the upcoming sections, we show how LineAO is efficiently evaluated by providing the definition of the zoom and visibility function in screen space as well as a proper weighting function g to ensure correct handling of thin structures locally and globally.

3.2 LineAO Evaluation in Screen Space

Until now, the LineAO sampling scheme was described in a general form. Admittedly, an evaluation in world space is not feasible, if real-time rendering is required. To solve this problem, we transfer the LineAO algorithm to screen space. To understand the following sections, we give a short overview on how screen space-based approaches work in general and which information is available in screen space. Furthermore, we define where LineAO is placed in this setting. In general, screen-space methods stand out as they process 3D scenes in two dimensions. They can be understood as advanced image filters. They render the scene to one or multiple 2D images and process these images on a per-pixel basis. Approaches in screen space have the advantage of knowing which part of the scene is visible and its relative depth with respect to the camera and projection setup. In other words, the rendering system provides a discretized representation of the scene. Besides this, additional information can be transported for each pixel, e.g., projected normals and camera information.

In the following section, we describe LineAO (7) as a function of each pixel P in the rendered line data set. We, therefore, replace the notation of *point* with *pixel*. We also provide a visibility function V and a special weighting function g for emphasizing high-frequency detail in narrow line structures and low-frequency features in a global context. As these functions work in screen space, they allow simplifications in visibility testing and weighting, which are not feasible in world space.

We list additional information needed by LineAO in screen space and where they can be gathered. Again, remember that each piece of information is available on a per-pixel basis; thus, we describe them as functions of a pixel P.

Projected normal $n_l(P)$**.** LineAO requires a normal at each point on the line. This was introduced in [13], where a line is interpreted as infinitesimally thin cylinder for normal calculation. We calculate the normal of a line during the rendering pass using its tangent T. With the vector C (pointing from the line point toward the camera), the normal in world space is defined as $\frac{T\times C}{|T\times C|}\times T$, which represents a vector pointing toward the camera that is perpendicular to the tangent. This normal is now projected using the projection matrix of the rendering pipeline. As we normalize the resulting vector, we omit the scaling by $\frac{1}{w}$ for dehomogenization as it is not needed. This yields a normal map $n_0(P)$ for the rendered lines. In addition, LineAO needs several l-times low-pass-filtered versions of this normal map. Therefore, we create a Gaussian pyramid $n_l(P)$ and call l the *Gauss level*.

Depth $d_l(P)$. During rendering, we store the depth value of each pixel in a depth map $d_0(P)$. Similar to the normal map, the depth map needs to be available in several low-pass-filtered versions. Thus, $d_l(P)$ denotes the l-times low-pass filtered, depth map for each pixel P. Again, l is called the *Gauss level*.

Zoom Level $z(P)$**.** If we assume a sampling sphere with a radius of one in world space, the sampling sphere in screen space (after projection) will have the radius $r'(P)$ for a pixel P and, therefore, represents the zooming factor applied by the projection. Using this as the zoom level $z(P) = r'(P)$ in (7) causes the hemisphere radii to adapt to the zoom level, thus ensuring a coherent LineAO effect when zooming.

With these fundamentals, we are now able to solve the LineAO equation in screen space.

3.2.1 Evaluating the Visibility Function

The visibility function V detects, whether a certain part on the hemisphere around a pixel P is occluded or not. Compared to other screen-space approaches such as [46], [47], we do not do a back projection to clip space for visibility checks. The idea is to interpret the depth map as a height field. A point in a valley is darkened due to the shadow a hill in its direct vicinity casts. With this metaphor in mind, we evaluate the occlusion term $1 - V_l(r\omega_i, P)$ from (6) by checking whether the pixel $P + r\omega_i$ on the sampling hemisphere is higher than P and, therefore, occluding P. Visibility is then defined by the discontinuous step function

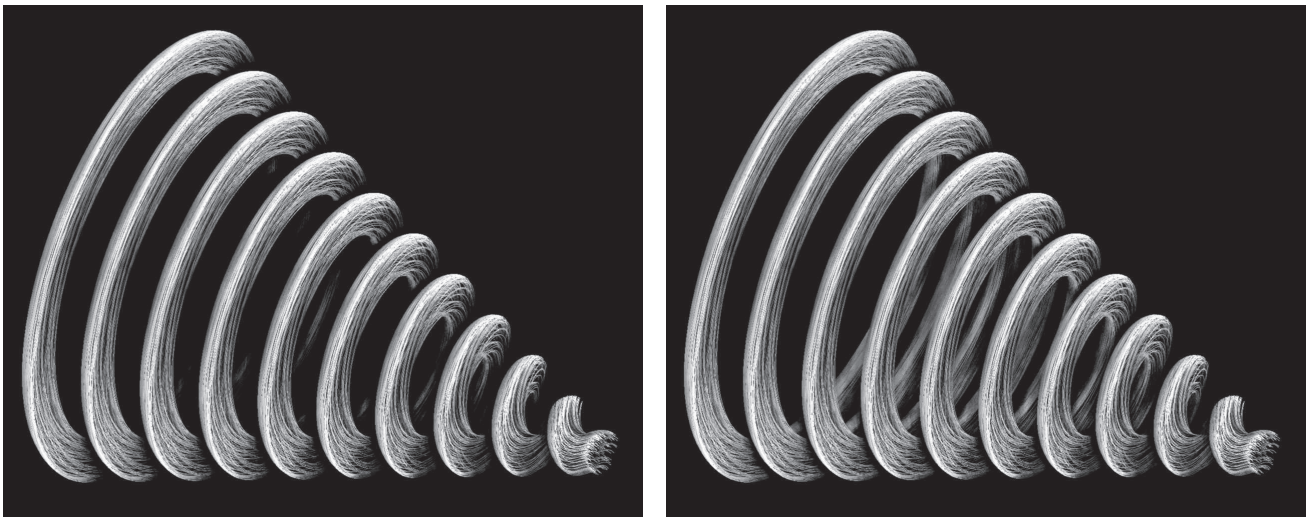
$$V_l(\omega, P) = \begin{cases} 1 & \text{if } d_l(P) - d_l(P+\omega) < 0 \\ 0 & \text{else,} \end{cases} \tag{8}$$

where a smaller depth value d_l indicates that the object is closer to the viewer.

By using the distance level as the Gauss level, dense line structures like bundles merge to solid geometry at higher distances and coarse or single lines disappear. In the next section, we introduce a very specialized weighting function which attenuates the visibility due to certain criteria, like distance, distance level l, and its surface properties to provide differentiated occlusion weighting for local detail and global structures.

3.2.2 Evaluating the Obscurance Weight

So far, we determined occlusion on a binary basis only. Either a pixel was occluded from one direction or not. The AO description from literature in (3) weights the binary occlusion by the diffuse reflection surface property. This models the real-world phenomenon of ambient skylight very well for solid geometry. For dense lines and other thin structures, we need a more sophisticated weighting scheme, which handles local structures, global structures,



(a) Constant $\delta(l) = 1$ (b) Our $\delta(l)$

Fig. 3. The influence of a proper falloff function for near and far occluders. In (a), a constant falloff is used. This creates high occlusion in line bundles but overestimates the occlusion of far away lines. In (b), our quadratic falloff is used. It creates high ambient occlusion in the spiral bundle while handling the more distant bundles properly.

and their interaction at once. This is not yet available in other SSAO approaches. Additionally, we include a weight to diminish the AO effect by the surface's lighting properties using the material's bidirectional reflectance distribution function (BRDF). This allows light sources to properly illuminate areas even if they are nearly invisible in terms of ambient occlusion, which cannot be compensated by applying lighting afterwards. This increases visibility of occluded structures if they are lit directly, which is very important in visualization.

To achieve this, we split the weighting into two parts: a depth-based attenuation of visibility and an attenuation of occlusion by the surface's lighting properties:

$$g_l(\omega, P) = g_l^{depth}(\omega, P) \cdot g_l^{light}(\omega, P). \tag{9}$$

Depth-based attenuation. For the depth-based attenuation, we use the same depth difference that was used for the visibility function:

$$\Delta d_l(\omega, P) = d_l(P) - d_l(P + \omega) \in [-1, 1]. \tag{10}$$

For near occluders, $\Delta d_l(\omega, P)$ is the occlusion intensity inside dense line bundles. For distant occluders, this describes the intensity of drop shadows and the inverse influence of empty space between several bundles. We need to define a falloff function $\delta(l)$ that attenuates the depth difference according to what kind of structure is currently sampled. Without a falloff, far occluders with a large depth difference would occlude each other too much, as shown in Fig. 3. Remembering that the parameter $l \in [0, s_r - 1]$ in (6) and (7) specifies whether local or global structures are sampled, the falloff is defined as

$$\delta(l) = \left(1 - \frac{l}{s_r}\right)^2 \in (0, 1]. \tag{11}$$

For near occluders, it is 1, generating a high occlusion for lines in proximity to others and converges toward 0 for far occluders. To additionally define the minimal depth difference needed to apply depth-based attenuation for occluder on P, we introduce the threshold $\delta_0 = 0.0001$. This now yields the depth-based attenuation as

$$g_l^{depth}(\omega, P) = \begin{cases} 0, & \text{if } \Delta d_l(\omega, P) > \delta(l) \\ 1, & \text{if } \Delta d_l(\omega, P) < \delta_0 \\ 1 - h\left(\dfrac{d_l(\omega, P) - \delta_0}{\delta(l) - \delta_0}\right), & \text{else.} \end{cases} \tag{12}$$

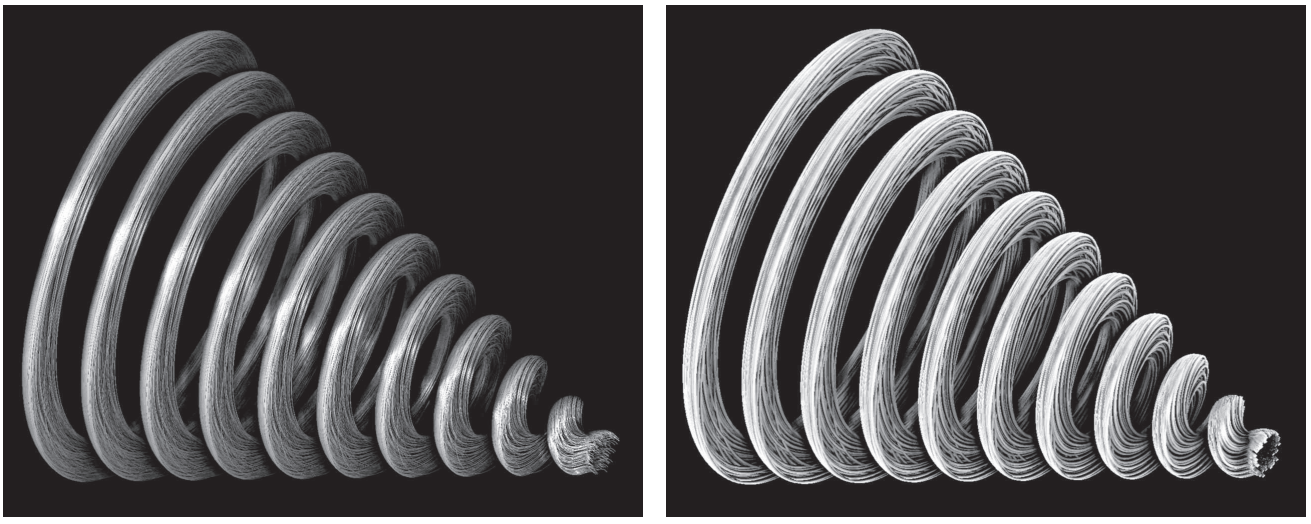
The value of g_l^{depth} is 1 for near occluders, whose depth difference to the current pixel is below δ_0, thus maximally emphasizing local structure in direct vicinity of the line. It is 0 and suppresses occlusion, if the depth difference exceeds the maximum defined by $\delta(l)$. For $l > 0$ (more distant occluders), this helps to avoid overly occluded distant line bundles as seen in Fig. 3. In between δ_0 and $\delta(l)$, we use the Hermite polynomial

$$h(x) = 3x^2 - 2x^3, \forall x \in [0, 1] : h(x) \in [0, 1] \tag{13}$$

on the depth difference scaled by the falloff function. For near occluders, g_l^{depth} emphasizes lines in direct vicinity of the line at P, thus emphasizing local structure in dense line bundles. For occluders near in the image plane but with a high depth difference, the occlusion effect is weakened to avoid heavy influence on the shading of the local structures. In Fig. 3b, this can be observed in the line bundles at the back side of the spiral. They still show the proper local structure without being darkened too much by the front side bundles as in Fig. 3a.

Illumination-based attenuation. By separating the calculation of LineAO and local illumination [13], dense and heavily occluded areas will be very dark even if these areas are directly lit by a source in their direct vicinity. To avoid this unwanted effect, we incorporate all the light sources of a scene into the LineAO calculation to attenuate the AO effect in these directly lit areas.

We define $BRDF(L_s, I_s, n, v)$ to be the illumination intensity of a light source s for a given light vector L_s, light intensity I_s, view vector v, and normal vector n. We



(a) LineAO using Illuminated Lines (b) LineAO on Tubes

Fig. 4. LineAO in combination with tube rendering (a) and illuminated lines (b). This yields additional cues for the shape of the line data and provides additional local structure detail.

calculate the reflected light L_l at the current point P in direction of ω using

$$L_l(\omega, P) = \sum_{s \in \text{Lights}} BRDF(L_s, I_s, n_l(P), \omega). \quad (14)$$

The lighting-based occlusion weight can then be defined as

$$g_l^{\text{light}}(\omega, P) = 1 - min(L_l(\omega, P), 1). \quad (15)$$

With this weight, we can attenuate the occlusion in brightly lit areas, with respect to the current sampling direction ω. As we did not normalize $L_l(\omega, P)$, we combine the influence of multiple lights on the local occlusion.

With the combination of depth- and light-based attenuation, the weighting function is able to emphasize local detail and their spatial relation as well as global structures without overemphasizing their shadowing effect (cf. Fig. 3). The depth-based attenuation seamlessly scales between local and global structures and, therefore, allows LineAO to create an AO-like effect for dense line renderings on multiple scales. The inclusion of light intensities ensures high visibility of local detail in otherwise occluded areas if they are directly lit.

3.2.3 LineAO Parameter Summary

In (7), we introduced three parameters. In this section, we have a closer look on these parameters, specific meaning, and their recommended values to achieve optimal results. Please note, that we have used these values for all images in this paper:

- $s_r = 3$—the number of radii to evaluate. This defines how many hemispheres are sampled around each pixel. The higher this value, the more detail influence the global AO effect. Evaluation at three radii ensures that detail in line bundles are rendered properly as well as smooth shadows of distant occluders. Higher values increase the total influence of smaller structures on the global AO effect. Lower values reduce visual quality, since many mid-range structures cannot be detected. Due to (11), s_r needs to be larger than 1.
- $s_h = 32$—the maximum number of samples on the hemisphere for each of the s_r iterations in LineAO. As we decrease the number of samples for increasing hemisphere radii as a harmonic series, the value of s_h defines the overall quality of the rendering with a tradeoff regarding render speed. In Section 4.1, we show several values of s_h in comparison. We found, that 32 is the best tradeoff between quality and speed, since higher values effect quality only marginally. According to (7), $s = 32$ and $s_r = 3$ yields in 59 samples being taken at each pixel.
- $r_0 = 1.5$ times the line width—the minimum radius. This radius defines the smallest hemisphere for sampling near occluders. This value defines how much local detail is incorporated into the global AO effect. To handle all local detail properly, this value needs to be close to the line width, defined by the rendering system. We chose 1.5 times the line width here, which ensures all local detail are preserved.

3.3 Combining LineAO with Other Methods

The LineAO approach can be easily combined with other approaches. When used with illuminated lines (Mallo et al. [13]), an additional shape cue is added: the specular highlight. This can be seen in Fig. 4a. As LineAO already represents the diffuse and ambient reflection, it is enough to additively combine the specular term of the illuminated lines approach with the LineAO intensity. When combining local illumination multiplicatively instead, an overly strong suppression of diffuse light will occur.

Besides illuminated lines, tube renderings are another interesting and commonly used alternative. In contrast to line renderings, tubes have a thickness in camera and in screen space. This is a remarkable difference from standard line primitives. Thus, we need to incorporate this in the parameter r_0 in (7). As the width t_s of a tube on the screen

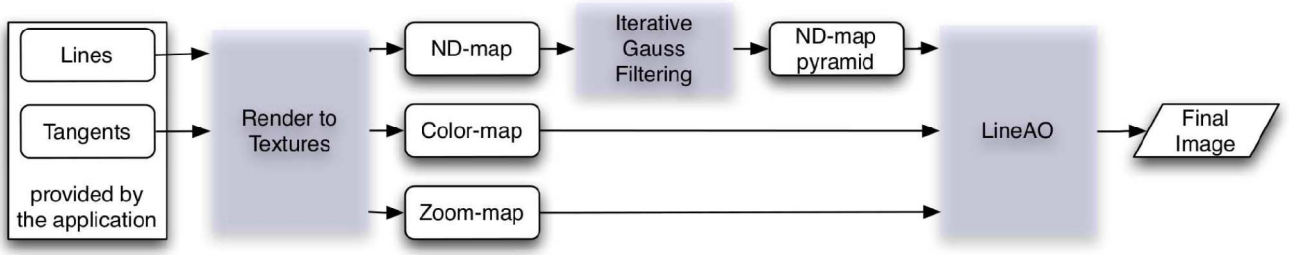


Fig. 5. The rendering pipeline. This figure shows the flow of information through the different rendering passes (gray boxes). The line and tangent data are provided by the application and uploaded to the GPU, where it is rendered to the ND map, color map, and zoom map. The ND map additionally gets processed on the GPU to create the needed s_r Gauss levels. The final rendering pass then applies the LineAO algorithm on a per-pixel basis and renders it to the screen.

varies depending on the zoom factor z, we have to define r_0 to be a function of z:

$$r_0^{\text{tubes}}(z) = 1.5 \cdot z \cdot t_s. \tag{16}$$

This is the definition of r_0 in Section 3.2.3, where the tube width is used instead of the line width. Fig. 4b shows the combination of LineAO with tube rendering [18].

LineAO is very flexible and allows combination with other lighting and shading schemes. Therefore, LineAO can be combined with any other line rendering methods easily.

3.4 Implementation

In the previous sections, we introduced our LineAO approach theoretically. In this section, we provide an implementation guideline using OpenGL and GLSL. Our LineAO implementation is available in the open-source visualization system *OpenWalnut* [59].

In Section 3.2, we gave an overview on how screen-space approaches work in general and which specific information LineAO needs beforehand. Fig. 5 shows our specific LineAO rendering pipeline and the flow of data between the consecutive rendering passes. We start by transferring the line data, including the tangent vector at each vertex, to the GPU. The standard OpenGL pipeline is utilized to render the scene to a texture using frame buffer objects (FBO). This creates a texture containing the plain rendered lines, including their coloring. For further reference, we call it *color map*. In this first step, we also calculate the unfiltered normal map $n_o(P)$, the depth $d_0(P)$ as well as the zoom level $z(P)$ using a fragment shader. Please note that the standard OpenGL pipeline does not calculate the depth value z but $\frac{z}{w}$, which is why we multiply the pixel's calculated depth by w to get the linear depth $d_0(P)$. We store the normal in the RGB-triple and the depth in the alpha value of an RGBA output texture. We call this texture *ND map* in the remaining section. The zoom level is written to a luminance texture, called *zoom map*.

After the first rendering pass, only the unfiltered ND map is available. From (7), one can see that we need $s_r - 1$ filtered versions of this map. This can be done on the GPU, using additional rendering passes. In these passes, a screen-filling quad is rendered to an FBO with half the resolution of the first one. For this quad, the ND map from the first pass is bound with mip-mapping enabled. This is done using the standard OpenGL pipeline and ensures that the half-resolution ND map, containing $n_1(P)$ and $d_1(P)$, is a low-pass-filtered version of $n_o(P)$ and $d_0(P)$. This is done iteratively $(s_r - 1)$-times and results in the s_r versions of the ND map.

The final render pass is the evaluation of the LineAO (7). We disable the FBO and render a screen-filling quad to the standard output framebuffer. As before, we bind the different levels of the ND map, the zoom map and the color map to this quad and apply a fragment program during rendering. This fragment program is now applied to each rendered fragment, which, in our case, equals the rendered pixels. We can now evaluate (7) for each P with a nested for-loop in GLSL. The LineAO parameters s_r, s_h, and r_0 are compile-time constants. This allows the GLSL compiler to unroll the loops and create optimized GPU code.

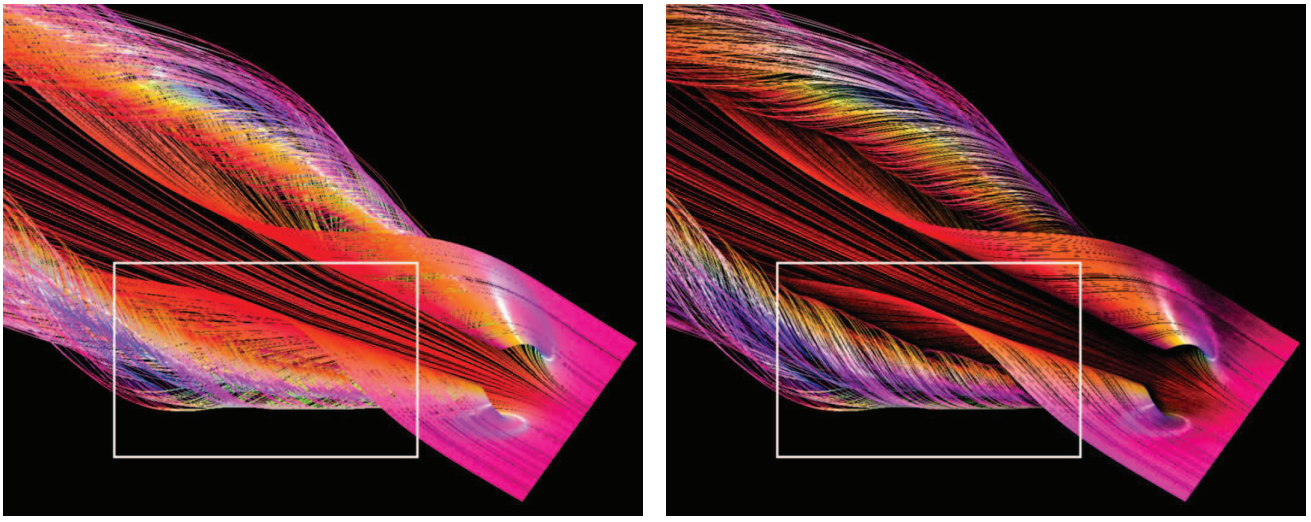
To avoid artifacts while sampling the hemisphere, we use a Monte Carlo sampling method. To achieve this, we create a low resolution, random vector map $R(P)$ and tile it on the quad to avoid upsampling in the LineAO pass and query two random vectors $R_1 = R(P)$ and $R_2 = R(\frac{i}{s}, \frac{j}{s_r - 1})$ for the current pixel P. The vector R_2 is queried at $(\frac{i}{s}, \frac{j}{s_r - 1})$, which is a point in the texture space, depending on current level and sample, according to (6) and (7). The sampling vector ω_i is then defined to be the random vector R_1 reflected along R_2. This avoids that the sampling vector ω_i is the same on different levels l of the LineAO equation and thus, avoids sampling in the same direction multiple times.

Another important thing to mention here is proper handling of boundary cases, e.g., if $P + r\omega$ (4) is outside the texture space. A simple approximation to handle this is by reflecting the vector $\vec{\omega}$ on the normal $n_l(P)$. Unfortunately, this causes overestimation if the area around the border is salient compared to the invisible area and underestimation in the opposite case. We solved this by rendering the scene slightly larger than the viewport, to provide the invisible but needed information for points close to the border of the viewport. This creates a coherent LineAO effect at the borders.

With this guideline, one is able to implement LineAO completely on the GPU. LineAO perfectly fits into the standard graphics pipeline and runs on consumer-level hardware.

4 Results

In the previous section, we have introduced our approach for improved shading and illumination of dense line data. LineAO, therefore, modifies and extends standard AO and SSAO, making it comply to the requirements and properties of line data. It phenomenologically creates the ambient



(a) Phong illuminated lines (b) LineAO with Phong illumination

Fig. 6. Streamlines around the main vortices of a delta wing data set obtained from a fluid dynamics simulation. (a) The specular highlights provide local shape information but are not able to properly represent the folding around the vortices. (b) LineAO enormously improves the perception of these folding structures.

occlusion effect, generating a more realistic image in real time with greatly increased spatial perceptibility.

Fig. 6 shows the streamlines around the two main vortices of a delta wing data set obtained from a fluid dynamics simulation. Especially, Fig. 6a does not provide any cues that allow derivation of streamline structure and depth. Our method adds these missing cues and depicts the folding around the main vortices much better.

Fig. 7 shows a fiber tractography data set of realistic size (74,313 lines containing a total of 11,000,000 vertices). It compares the LineAO rendered data set with plain illuminated lines rendering, standard SSAO approach from Crytec [46], and ray tracing using the ray tracer package POV-Ray [60]. Although the plain illuminated line rendering does not provide any spatial cues and only little shape hint due to local specular highlights, it is still the standard way of exploring large medical data like this. Only interaction can reveal further structural information. Fig. 7b shows a standard SSAO algorithm applied to the same line data. Although it reveals some global structure with a halo effect, the technique generally is not suited for line data. This can be seen in the underestimated thin structures on top of the image as well as on the overestimated fissures of the *Frontal Lobe*. Increasing the number of samples in SSAO does not solve this problem, as it is still not able to distinguish local and distant occluders properly. For comparison, we also rendered a ray-traced image, with POV-Ray's radiosity approach enabled. As it is not possible to ray-trace lines per se, we have used thin, arithmetically described cylinders with spheres as joints to describe the line strips. Besides the enormous calculation time of 14 hours, the POV-Ray renderer suffers the problem of intense self-shadowing in dense areas, causing overlapping and dense bundles to be undistinguishable. With LineAO, it is possible to distinguish individual lines and bundles as well as their layering even without interaction. It is possible to see the structure of the fissures in the *Frontal Lobe* and the layering of bundles in the brain stem.

In Fig. 8, a selected part of a brain fiber tractography data set is shown. The LineAO approach in combination with tube rendering shows spatial relations of these bundles in a very natural way. Even distinguishing a single fiber in a bundle is possible. Without LineAO, estimation of vicinity between these three selected bundles is difficult and bundle shape can only be recognized with interaction.

4.1 Performance and Accuracy

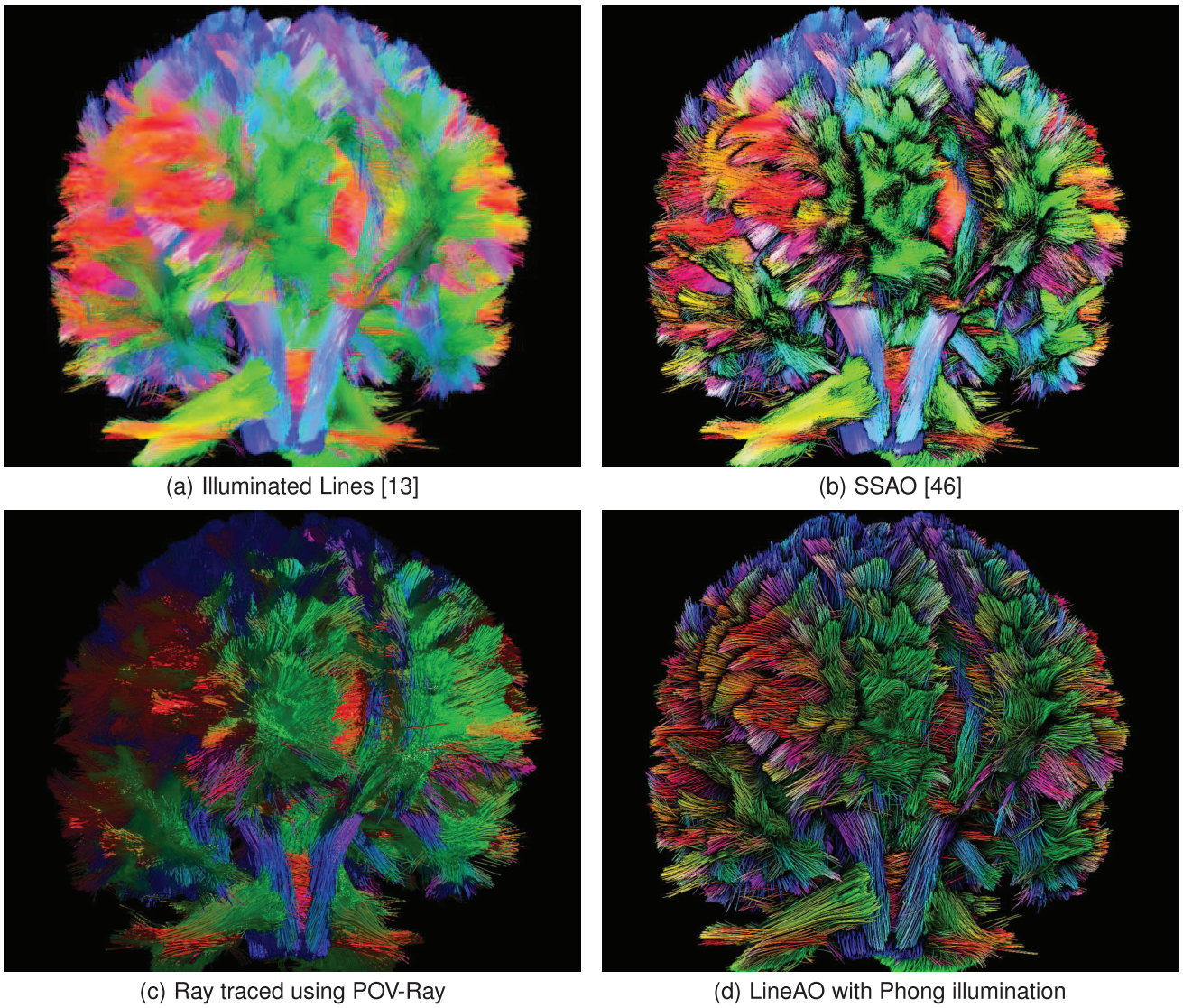
In Section 3.2.3, we listed the parameters and our default settings. We have used these values for rendering all images throughout the paper. Thereby, the sampling count s influences the overall rendering performance and a tradeoff between accuracy and performance has to be done. Fig. 9 compares a drastically zoomed part of the delta wing vortices data set, rendered with different sample counts. It clearly shows that sample counts lower than 32 introduce significant noise artifacts, whereas values above 32 only slightly improve rendering quality while diminishing rendering performance drastically.

To measure LineAO performance, we used a machine with two Quad-Core AMD Opteron 2,352 processors, 32 GB RAM, and a GeForce GTX480 graphics card. However, the CPU and RAM do not play an important role here since the computation is done on the GPU only. Table 1 compares the frames per second (FPS) of LineAO with plain Phong illumination at a screen resolution of $1{,}280 \times 1{,}024$ pixel.

With this, it is evident that Phong illumination is data-bound, whereas LineAO and SSAO mainly depend on the number of pixels covered by the rendered lines. Phong is evaluated for each fragment instead of each pixel, as LineAO does. This explains why larger data set sizes do not have a drastic impact on FPS for the additional LineAO pass, compared to Phong illuminated lines.

4.2 Limitations

The main disadvantage and limitation is that LineAO does not provide improved shading for very coarse data. The idea behind ambient occlusion is to model diffuse skylight and the occlusion of geometry by other geometry (the occluders). In our case, dense line structures provide a high occlusion inside the bundles and smooth shadows for



(a) Illuminated Lines [13] (b) SSAO [46]

(c) Ray traced using POV-Ray (d) LineAO with Phong illumination

Fig. 7. Comparison of different line rendering approaches with a deterministic fiber tracking of the human brain. The fibers are colored using the tangent-based directional colormap [8], which is very common in the neurosciences. A prominent example for comparing each technique is the layering of bundles at the brain stem, where the *Pons* and *Medulla Oblongata* pass into the *Spinal Cord*. In comparison to (a), (b), and (c), LineAO (d) is able to show the local structure of fibers in a bundle, as well as their spatial relation in a global scope, without intense self-shadowing in dense and overlapping areas. Besides ray tracing, all methods perform in real time. (c) Took 14h to compute.

salient structures. Very coarse lines and thin structures naturally do not cause much occlusion. To overcome this problem to a certain degree, the line structures have to be thickened. One solution to this problem is to use a higher line width or tube renderings, as shown in Fig. 8b.

5 Conclusion

With current line rendering methods, it is not sufficiently possible to distinguish associated bundles of lines and their spatial relation to others locally and globally. Thus, we proposed a novel approach, tailored toward line rendering, provided that global *and* local structural and spatial information is retained. It allows to grasp the structure of bundles and the spatial relation between structures at local *and* global scope. The combination with directed local illumination adds further perceptual cues for the local characteristics of single lines and line bundles. Its simplicity and performance distinguishes it from other approaches, which use complex, surface-based techniques or expensive precalculations to provide insight into structures. Our presented method does not require any precomputation and does not introduce additional geometry for each line segment. The real-time ability and the possibility of combining LineAO with other interactive methods allows it to be used as add-on to existing approaches in interactive and explorative visualization environments.

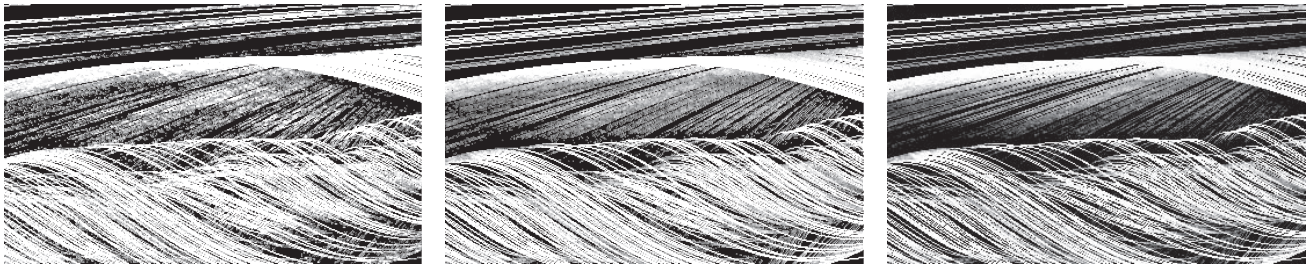
Our approach is a considerable step toward visual quality, spatial perception, and usefulness of line-based visualization techniques and perfectly fits into the constraints of modern, interactive data exploration and visualization environments.

6 Future Work

The limitation of AO toward dense and compact line data is a major drawback of this principle and, therefore, affects

(a) Illuminated tubes (b) LineAO with Phong illuminated tubes

Fig. 8. This picture shows the combination of local illumination, tubes, and LineAO in a part of the Corpus Callosum (red), Cingulum (green), and Corticospinal tract (blue). The illuminated tubes already provide some structural information for each line but make it hard to distinguish individual lines. Their spatial relation is not completely visible. For example, the shape of the Corticospinal Tract (blue) is not fully determined with illuminated tubes and seems to be a round bundle. With LineAO, it is immediately clear, that this tract has a flat shape. Besides this, its distance to the Corpus Callosum (red) cannot be estimated with illuminated tubes, whereas LineAO reveals their closeness.



(a) $s = 8$ — 63FPS (b) $s = 32$ — 37FPS (c) $s = 64$ — 24FPS

Fig. 9. Top view of the delta wing vortices, zoomed into the part framed in Fig. 6 to show quality difference between the different sample counts. (a) contains a lot of noise artifacts compared to (b), whereas the visual difference between (b) and (c) is insignificant.

LineAO as well. The next step is to combine LineAO with halo-based techniques, which are able to provide depth cues for coarse line structures. Another major goal is to adaptively sample the occlusion integral, depending on density information. Although this would currently involve an additional precomputation step, we are aiming toward an estimation of density of lines in screen space in real time.

7 Acknowledgments

The authors thank the Max Planck Institute for Human Cognitive and Brain Sciences Leipzig for providing the human brain data set and Markus Rütten, DLR in Göttingen, for providing the delta wing data set. They thank the members of the Image and Signal Processing Group, University of Leipzig, Germany, and the OpenWalnut development team for their comments and support.

TABLE 1
Performance of LineAO for Several Data Sets in Comparison to Plain Phong Illuminated Lines and SSAO [46]

Figure	Line strips	Vertices	FPS Phong	FPS SSAO	FPS LineAO
4(b)	1 000	1 000 000	140	42	17
4(a)	1 000	1 000 000	460	59	37
6	1 000	3 600 000	260	55	20
7(d)	74 313	11 000 000	30	22	16
8	564	102 700	500	80	30

References

[1] T. Delmarcelle and L. Hesselink, "Visualizing Second-Order Tensor Fields with Hyperstreamlines," *IEEE Computer Graphics Applications*, vol. 13, no. 4, pp. 25-33, July 1993.

[2] G. Reina, K. Bidmon, F. Enders, P. Hastreiter, and T. Ertl, "GPU-Based Hyperstreamlines for Diffusion Tensor Imaging," *Proc. EUROGRAPHICS/IEEE VGTC Symp. Visualization*, pp. 35-42, 2006.

[3] P.J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi, "In Vivo Fiber Tractography Using Dt-Mri Data," *Magnetic Resonance in Medicine: Official J. Soc. of Magnetic Resonance in Medicine/Soc. of Magnetic Resonance in Medicine*, vol. 44, no. 4, pp. 625-632, 2000.

[4] S. Mori and P.C.M. van Zijl, "Fiber Tracking: Principles and Strategies - a Technical Review," *NMR in Biomedicine*, vol. 15, nos. 7/8, pp. 468-480, 2002.

[5] J. Berman, M. Berger, P. Mukherjee, and R. Henry, "Diffusion-Tensor Imaging-Guided Tracking of Fibers of the Pyramidal Tract Combined with Intraoperative Cortical Stimulation Mapping in Patients with Gliomas," *Neurosurgery*, vol. 101, no. 1, pp. 66-72, 2004.

[6] M. Hlawitschka and G. Scheuermann, "HOT-Lines—Tracking Lines in Higher Order Tensor Fields," *Proc. IEEE Visualization Conf.*, C.T. Silva, E. Gröller, and H. Rushmeier, eds., pp. 27-34, 2005.

[7] T. Schultz and H.-P. Seidel, "Estimating Crossing Fibers: A Tensor Decomposition Approach," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1635-1642, Nov./Dec. 2008.

[8] S. Pajevic and C. Pierpaoli, "Color Schemes to Represent the Orientation of Anisotropic Tissues from Diffusion Tensor Data: Application to White Matter Fiber Tract Mapping in the Human Brain," *Magnetic Resonance in Medicine,* vol. 43, no. 6, pp. 921-921, 2000.

[9] L. Wanger, "The Effect of Shadow Quality on the Perception of Spatial Relationships in Computer Generated Imagery," *Proc. Symp. Interactive 3D Graphics (I3D '92),* pp. 39-42, 1992.

[10] L.C. Wanger, J.A. Ferwerda, and D.P. Greenberg, "Perceiving Spatial Relationships in Computer-Generated Images," *IEEE Computer Graphics Applications,* vol. 12, no. 3, pp. 44-51, May 1992.

[11] V.S. Ramachandran, "Perception of Shape from Shading," *Nature,* vol. 331, pp. 163-166, 1988.

[12] M. Langer and H. Bülthoff, "Depth Discrimination from Shading under Diffuse Lighting," *Perception,* vol. 29, pp. 649-660, 2000.

[13] O. Mallo, R. Peikert, C. Sigg, and F. Sadlo, "Illuminated Lines Revisited," *Proc. IEEE Visualization Conf.,* pp. 19-26, 2005.

[14] M. Zöckler, D. Stalling, and H.-C. Hege, "Interactive Visualization of 3d-Vector Fields Using Illuminated Stream Lines," *Proc. Seventh Conf. Visualization (VIS '96),* pp. 107-113, 1996.

[15] J.F. Blinn, "Models of Light Reflection for Computer Synthesized Pictures," *Proc. SIGGRAPH '77,* pp. 192-198, 1977.

[16] C. Stoll, S. Gumhold, and H.-P. Seidel, "Visualization with Stylized Line Primitives," *Proc. IEEE Visualization Conf. (VIS '05),* C.T. Silva, E. Gröller, H. Rushmeier, eds., pp. 695-702, 2005.

[17] M. Schirski, T. Kuhlen, M. Hopp, P. Adomeit, S. Pischinger, and C. Bischof, "Efficient Visualization of Large Amounts of Particle Trajectories in Virtual Environments Using Virtual Tubelets," *Proc. ACM SIGGRAPH Int'l Conf. Virtual Reality Continuum and Its Applications in Industry (VRCAI '04),* pp. 141-147, 2004.

[18] D. Merhof, M. Sonntag, F. Enders, C. Nimsky, P. Hastreiter, and G. Greiner, "Hybrid Visualization for White Matter Tracts Using Triangle Strips and Point Sprites," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1181-1188, Sept. 2006.

[19] M.H. Everts, H. Bekker, J.B. Roerdink, and T. Isenberg, "Depth-Dependent Halos: Illustrative Rendering of Dense Line Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 1299-1306, Nov./Dec. 2009.

[20] G. Schussman and K.-L. Ma, "Anisotropic Volume Rendering for Extremely Dense, Thin Line Data," *Proc. Conf. Visualization (VIS '04),* pp. 107-114, 2004.

[21] C. Yuksel and S. Tariq, "Advanced Techniques in Real-Time Hair Rendering and Simulation," *Proc. ACM SIGGRAPH,* pp. 1-168, 2010.

[22] K. Ward, F. Bertails, T.-Y. Kim, S.R. Marschner, M.-P. Cani, and M.C. Lin, "A Survey on Hair Modeling: Styling, Simulation, and Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 2, pp. 213-234, Mar. 2007.

[23] L. Petrovic, M. Henne, and J. Anderson, "Volumetric Methods for Simulation and Rendering of Hair," Pixar Technical Memo 06-08, pp. 1-6, 2006.

[24] C. Yuksel and E. Akleman, "Rendering Hair with Global Illumination," *Proc. ACM SIGGRAPH,* p. 124, 2006.

[25] C. Yuksel, E. Akleman, and J. Keyser, "Practical Global Illumination for Hair Rendering," *Proc. Pacific Conf. Computer Graphics and Applications,* 2007.

[26] M. Ruiz, I. Boada, I. Viola, S. Bruckner, M. Feixas, and M. Sbert, "Obscurance-Based Volume Rendering Framework," *Proc. IEEE/EG Int'l Symp. Vol. and Point-Based Graphics,* pp. 113-120, 2008.

[27] J. Kronander, D. Jonsson, J. Low, P. Ljung, A. Ynnerman, and J. Unger, "Efficient Visibility Encoding for Dynamic Illumination in Direct Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 18, no. 3, pp. 447-462, Mar. 2012.

[28] V. Šoltészová, D. Patel, S. Bruckner, and I. Viola, "A Multidirectional Occlusion Shading Model for Direct Volume Rendering," *Computer Graphics Forum,* vol. 29, no. 3, pp. 883-891, 2010.

[29] C. Wyman, S. Parker, P. Shirley, and C. Hansen, "Interactive Display of Isosurfaces with Global Illumination," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 2, pp. 186-196, Mar. 2006.

[30] Z. Melek, D. Mayerich, C. Yuksel, and J. Keyser, "Visualization of Fibrous and Thread-Like Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1165-1172, http://dx.doi.org/10.1109/TVCG.2006.197, Sept./Oct. 2006.

[31] M. Tarini, P. Cignoni, and C. Montani, "Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1237-1244, Sept. 2006.

[32] M. Krone, K. Bidmon, and T. Ertl, "Interactive Visualization of Molecular Surface Dynamics," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 1391-1398, Nov./Dec. 2009.

[33] C.P. Gribble and S.G. Parker, "Enhancing Interactive Particle Visualization with Advanced Shading Models," *Proc. Third Symp. Applied Perception in Graphics and Visualization (APGV '06),* pp. 111-118, http://doi.acm.org/10.1145/1140491.1140514, 2006.

[34] O. Arikan, D.A. Forsyth, and J.F. O'Brien, "Fast and Detailed Approximate Global Illumination by Irradiance Decomposition," *Proc. ACM SIGGRAPH,* pp. 1108-1114, 2005.

[35] P. Shanmugam and O. Arikan, "Hardware Accelerated Ambient Occlusion Techniques on Gpus," *Proc. Symp. Interactive 3D Graphics and Games (I3D '07),* pp. 73-80, 2007.

[36] C. Reinbothe, T. Boubekeur, and M. Alexa, "Hybrid Ambient Occlusion," *Proc. EUROGRAPHICS Conf.,* pp. 51-57, 2009.

[37] M. McGuire, "Ambient Occlusion Volumes," *Proc. ACM Conf. High Performance Graphics,* pp. 47-56, 2010.

[38] S. Zhukov, A. Inoes, and G. Kronin, "An Ambient Light Illumination Model," *Proc. Eurographics Workshop Rendering Techniques,* G. Drettakis and N. Max, eds., pp. 45-56, 1998.

[39] D.R. Baum, H.E. Rushmeier, and J.M. Winget, "Improving Radiosity Solutions through the Use of Analytically Determined Form-Factors," *Proc. 16th Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 325-334, 1989.

[40] M. Bunnell, "Dynamic Ambient Occlusion and Indirect Lighting," *GPU Gems 2,* Chapter 14, Addison-Wesley Professional, 2005.

[41] J. Hoberock and Y. Jia, "High-Quality Ambient Occlusion," *GPU Gems 3,* Chapter 12, Addison-Wesley Professional, 2005.

[42] Z. Ren, R. Wang, J. Snyder, K. Zhou, X. Liu, B. Sun, P.-P. Sloan, H. Bao, Q. Peng, and B. Guo, "Real-Time Soft Shadows in Dynamic Scenes Using Spherical Harmonic Exponentiation," *Proc. ACM SIGGRAPH,* pp. 977-986, 2006.

[43] R. Wang, K. Zhou, J. Snyder, X. Liu, H. Bao, Q. Peng, and B. Guo, "Variational Sphere Set Approximation for Solid Objects," *Visual Computer,* vol. 22, pp. 612-621, 2006.

[44] G. Bradshaw and C.O' Sullivan, "Sphere-Tree Construction Using Dynamic Medial Axis Approximation," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation,* pp. 33-40, 2002.

[45] T. Luft, C. Colditz, and O. Deussen, "Image Enhancement by Unsharp Masking the Depth Buffer," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 1206-1213, 2006.

[46] M. Mittring, "Finding Next Gen: Cryengine 2," *Proc. ACM SIGGRAPH,* pp. 97-121, 2007.

[47] V. Kajalin, "Screen Space Ambient Occlusion," *ShaderX7: Advanced Rendering Techniques,* pp. 413-424, Charles River Media, 2009.

[48] L. Bavoil and M. Sainz, "Multi-Layer Dual-Resolution Screen-Space Ambient Occlusion," *Proc. SIGGRAPH,* pp. 45:1-45:1, 2009.

[49] D. Filion and R. McNaughton, "Effects & Techniques," *Proc. ACM SIGGRAPH,* pp. 133-164, 2008.

[50] T.-D. Hoang and K.-L. Low, "Multi-Resolution Screen-Space Ambient Occlusion," *Proc. Computer Graphics Int'l Conf. (CGI '11),* 2011.

[51] A. Evans, "Fast Approximations for Global Illumination on Dynamic Scenes," *Proc. ACM SIGGRAPH,* pp. 153-171, 2006.

[52] J. Kontkanen and S. Laine, "Ambient Occlusion Fields," *Proc. Symp. Interactive 3D Graphics and Games (I3D '05),* 2005.

[53] M. Malmer, F. Malmer, U. Assarsson, and N. Holzschuch, "Fast Precomputed Ambient Occlusion for Proximity Shadows," *J. Graphics Tools,* vol. 12, no. 2, pp. 59-71, 2007.

[54] D. Lacewell, B. Burley, S. Boulos, and P. Shirley, "Raytracing Prefiltered Occlusion for Aggregate Geometry," *Proc. IEEE Symp. Interactive Raytracing,* 2008.

[55] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments," *Proc. 29th Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 527-536, 2002.

[56] P.-P. Sloan, "Normal Mapping for Precomputed Radiance Transfer," *Proc. Symp. Interactive 3D Graphics and Games (I3D '06),* pp. 23-26, 2006.

[57] D.L. James and K. Fatahalian, "Precomputing Interactive Dynamic Deformable Scenes," *ACM Trans. Graphics,* vol. 22, pp. 879-887, 2003.

[58] T.-D. Hoang and K.-L. Low, "Multi-Resolution Screen-Space Ambient Occlusion," *Proc. Int'l Conf. Computer Graphics,* http://www.comp.nus.edu.sg/~duong/, 2011.
[59] "OpenWalnut,"Ternet, http://www.openwalnut.org, 2012.
[60] P. of Vision Pty. Ltd, "Pov-Ray Raytracer,"ternet, http://www.povray.org, 2012.

Sebastian Eichelbaum received the MS degree (Diplom) in computer science in 2009 from the Universität Leipzig, Germany. He is currently working toward the PhD degree in the Department of Computer Science at the Universität Leipzig, where his research focuses on visualization in neurosciences and computer graphics. He is a member of the IEEE Computer Society.

Mario Hlawitschka studied computer sciences and electrical engineering with a focus on signal processing and visualization, and received the BSc and MSc (Diplom-Informatiker) degrees from the University of Kaiserslautern, Germany, in 2004. He received the PhD (Dr. rer. nat.) degree from the Universität Leipzig, Germany in 2008 for his work on visualization of medical data, and completed postdoctoral research in the Department of Biomedical Engineering and the Department of Computer Sciences at the University of California, Davis. He is currently a professor for scientific visualization at the Department of Computer Science at the Universität Leipzig. He is a member of the IEEE Computer Society.

Gerik Scheuermann received the BS and MS degrees in mathematics in 1995 from the University of Kaiserslautern where he received the PhD degree in computer science in 1999. During 1995-1997, he conducted research at Arizona State University for about a year. He worked as postdoctoral researcher at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis, in 1999 and 2000. Between 2001 and 2004, he was an assistant professor for computer science at the University of Kaiserslautern. Currently, he is a full professor in the Computer Science Department at the University of Leipzig. His research topics include algebraic geometry, topology, Clifford algebra, image processing, graphics, and scientific visualization. He is a member of the ACM, IEEE, and GI.