Illustrative Rendering of Dense Line Data

Maarten H. Everts, Henk Bekker, Jos B.T.M. Roerdink, and Tobias Isenberg

Abstract—We present a technique for the illustrative rendering of 3D line data at interactive frame rates. We create depth-dependent halos around lines to emphasize tight line bundles while less structured lines are de-emphasized. Moreover, the depth-dependent halos combined with depth cueing via line width attenuation increase depth perception, extending techniques from sparse line rendering to the illustrative visualization of dense line data. We demonstrate how the technique can be used, in particular, for illustrating DTI fiber tracts but also show examples from gas and fluid flow simulations and mathematics.

Keywords—Illustrative rendering and visualization, NPR, dense line data, DTI, black-and-white rendering, GPU technique.

1 Introduction

Illustrative depictions play an essential role in the communication of knowledge. Traditionally, illustrators used graphic tools such as pen-and-ink to draw—with the goal to depict, e. g., the shapes of objects. The choice of tool was typically dictated by the means of reproduction, usually the printing in books. Therefore, illustrators often chose techniques that result in black-and-white imagery, for example pen-and-ink, woodcuts, or copper plates. Despite being limited to two "colors," these techniques still allowed illustrators to convey shape, material, and illumination through techniques such as stippling, hatching, or cross-hatching. More importantly, illustrators also made use of the fundamental illustration principles of *abstraction* and *emphasis* to effectively communicate their intentions.

With the advance of computer support, illustrators started to use general purpose graphics programs (e. g., Adobe's Illustrator™) to create illustrations. One reason for this tool change is that general purpose programs, in many aspects, provide more freedom than traditional tools. In a separate development, the visualization community has created numerous successful techniques to solve specialized visualization problems, e. g., in the medical domain. In both cases, the availability of color processing and reproduction has invited the use of shading techniques, i. e., representing surfaces through shades of color, in contrast to the traditional black-and-white methods.

While illustrators in their use of general purpose tools can still apply the illustration principles of abstraction and emphasis, this is more difficult for automatic techniques as it is challenging to "teach" importance to an algorithm. In the areas of non-photorealistic rendering (NPR) and illustrative visualization, however, abstraction and emphasis techniques have been investigated. Examples include the use of halos for simple line rendering [2, 10] or shading [20, 33] and volume rendering [4], the use of additional depth cueing by influencing line or shading parameters (e. g., [10]), interactive emphasis techniques (e. g., [30, 37]), or focus+context techniques (e. g., [12, 35]).

In this paper we build on these previous approaches but focus on a specific subset of data—line datasets (see the example result in Fig. 1). This type of data is generated in a number of application domains such as medical imaging (e. g., DTI fiber tract extraction), meteorology (e. g., particle traces in storm data or simulations), physics (e. g., particle tracts from 3D gas or fluid flow simulations), or astronomy (e. g., particle traces from mass distribution simulations for galaxy formation). In all these application areas, line data with a comparably high density of elements is generated and needs to be analyzed. This data lends itself more to the traditional black-and-white depiction techniques rather than shading-based methods because lines occupy much less space than shaded elements such as cylindrical shapes. In addition, detail in the visualizations is often important so that reducing the number of depicted lines may not be a suitable approach.

Fig. 1. Illustrative visualization of DTI fiber tracts with depth-dependent halos.

To address this problem of depicting dense line datasets in their full detail our paper makes the following contributions: We show how to illustratively visualize dense line datasets at interactive frame-rates using modern graphics hardware. We introduce a conceptually simple technique that allows us to only render the front layer of the data, i. e., the lines or points that lie close to each other and closest to the viewer. These front elements are rendered such that they do not overlap each other but at the same time they occlude elements much further away from the viewer. This *depth-dependent halo* technique emphasizes bundles of co-linear line segments (which are likely to be important, see Fig. 1) and abstracts from less structured segments. Moreover, we de-emphasize elements that are farther away to enhance depth perception, and filter the dataset for further emphasis of important structures. We further show how the discrete and black-and-white nature of the depicted elements lends itself to anaglyphic stereo rendering.

The remainder of the paper is organized as follows. In Section 2 we place our work in the context of related approaches.

Next, we present the technical details of the approach in Section 3. Then in Section 4, we show examples of visualizations created with our technique for several application domains. We conclude the paper in Section 5, where we also mention some avenues for future work.

2 Related Work

As background for our illustrative line rendering we discuss both line visualization and illustrative rendering techniques.

2.1 Line Data Visualization

Numerous techniques exist to depict paths of particles or other linear structures, in particular for flow visualization. For example, people have employed (shaded) lines, tubes, or strips (ribbons) whose color and shape can be changed depending on data properties such as velocity, flux, or direction [26] (e. g., Fig. 2(a)). Particularly related to our work are scalable, self-orienting surface techniques [21, 22, 28, 29] which create shaded, view-aligned strips to visualize 3D vector fields. In contrast, our goal is to create high-resolution black-and-white visualizations for dense datasets using illustration principles.

As an alternative to explicitly representing streamlines, texture-based methods [19] such as line integral convolution [6] provide both a global and local impression of flow data. This technique was extended to 3D [13, 15] where, related to our work, halos are used to increase depth perception. For a similar purpose, halos are used in streamline-based volume visualization [36]. Both streamlines and texture-based techniques are employed in many other domains which rely on the visualization of line data resulting from real or simulated linear structures or particle traces. Examples include physics (e. g., electric or magnetic field lines), chemistry (e. g., protein structures), and meteorology (e. g., storm data).

In particular in the medical domain, line data such as tracts of muscle or brain fibers is important. Here, fiber tracts are estimated from diffusion weighted magnetic resonance imaging (DW-MRI) [23]. The fiber tracts represent, for example, bundles of neural axons connecting different parts of the brain. Such fiber tracts are typically rendered as lines or tubes with coloring or shading applied to them to enhance understanding of spatial relationships [25, 39, 40]. The approach we present in this paper uses illustration principles to create black-and-white visualizations that despite being limited to two colors still convey the spatial relationships of the lines.

2.2 Illustrative Visualization and Rendering

In a number of line-based scientific visualization techniques, people make use of illustration principles. For instance, Joshi et al. [16] employ techniques that enhance the boundary or silhouette to accentuate internal features in visualizations of hurricanes. Similar contour enhancing techniques have also been investigated for flow data [31] or medical volume data [5, 9]. Related to illustrative visualization is non-photorealistic rendering (NPR), where lines have been used as a means to illustrate surfaces (e. g., [14, 38]) but are usually placed onto surfaces during rendering rather than being the original carrier of the depicted data or shape. Such techniques have been applied, e. g., to medical volume visualization [8, 24, 34] as well as rendering of fiber and vessel structures [18, 27]. In the latter examples, line rendering is used to enhance and supplement traditional techniques that are based on larger cylindric structures.

Techniques that enhance depth perception are important specifically in line rendering and have been introduced to NPR in its early days. Such techniques include the use of visibility information [1, 17] as well as the use of halos [2, 10], the illustration method we also use in our own work. Halos, however, can not only be used in line rendering but have been applied in more traditional visualization techniques based, e. g., on line integral convolution [15] or volume rendering [4] to enhance depth perception. Related to these approaches as well as to our own are techniques that make use of depth buffer manipulations to enhance the depth perception in the created images. Noteworthy in this respect are, in particular, depth buffer unsharp masking [20], depth cueing in molecular visualization [33], and pen-and-ink tree rendering using depth discontinuities [7]. In contrast to these techniques, our approach extends similar illustrative rendering principles to the domain of dense line or point data which are rendered without shading and which, thus, rely even more on cues to indicate depth relations.

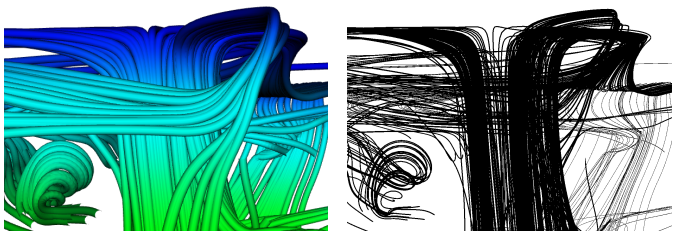
3 Illustrative 3D Line Rendering

Based on the previously discussed techniques to visualize line data, we combine these with principles of halo-based non-photorealistic rendering of lines. We focus on datasets where dense sets of lines are important, for example, DTI fiber tracts or particle traces in physical simulations. In this section we first give a general motivation and overview of the technique and then discuss its realization in detail. Next, we show how the technique is extended to point clouds, present a number of visual enhancements, and address data filtering.

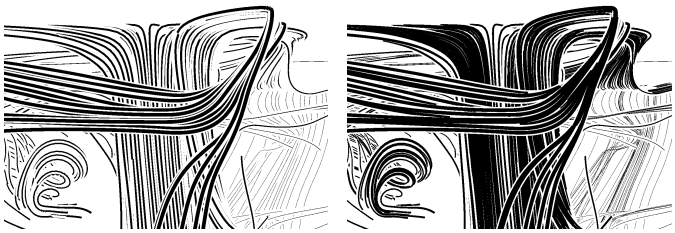
3.1 General Motivation and Technique Overview

The rendering of line data requires, in particular, that the depth relation of the lines is clearly depicted. As discussed in the previous section, shaded cylindrical representations were traditionally employed for this purpose. This approach has a number of limitations. Due to the use of shading, each line needs to have a certain minimum width in order for viewers to be able to discriminate each individual line's orientation and location in space (Fig. 2(a)). This limits the number of lines that can simultaneously be depicted and also hinders the visualization of natural line bundles if each line is to be visible individually. An alternative approach would be to use simple black lines on a white background (Fig. 2(b)). On the one hand, a higher number of data elements can potentially be depicted because lines can easily be packed more tightly. Moreover, it also becomes more feasible to use such illustrations in print, because these do not require shading and thus do not rely on halftoning. On the other hand, this introduces a lot of visual clutter into the image and results in large regions of black being shown. Thus, it is no longer possible to distinguish foreground from background lines, which also means that the depth relation is lost.

To address these issues and to be able to use the advantages from both techniques, we employ line halos as previously used in line rendering [2, 10]. Since we are dealing with dense

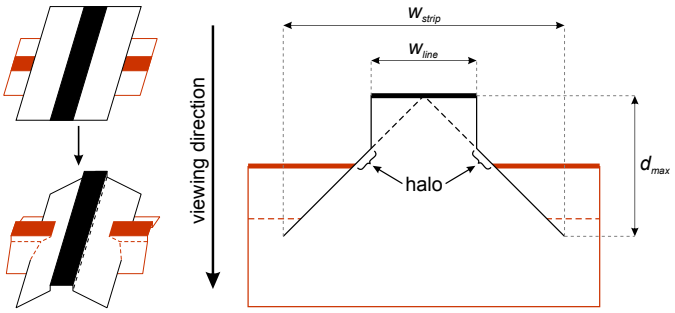


(a) Traditional tubes with shading. (b) Plain line rendering.



(c) All lines with halos. (d) Lines with illustrative halos.

Fig. 2. Comparison of rendering techniques for line datasets.



(a) (b)

Fig. 3. Schematic depiction of the rendering of the creation of depth-dependent halos for large depth discontinuities: (a) projected view of the depth displacement, (b) view of a line (black) being crossed by a perpendicular line (red) creating a halo. The fat parts are the actually rendered line pixels, the thin lines illustrate the z-buffer manipulation.

datasets of lines, simply adding a halo to each line is not sufficient (Fig. 2(c)). Instead, we present a technique that adds halos to all lines, but these halos are only rendered if the lines are sufficiently separated in depth (Fig. 2(d)). This causes the halos of lines that lie in front of others to occlude lines further away. If lines have the same distance to the viewer, however, they do not occlude one another. This *depth-dependent halo* reduces visual clutter, emphasizes line bundles by visually clustering them, and depicts depth relations as in previous halo techniques [2, 4, 10], resulting in an effective illustration.

The general approach is to use view-aligned triangle strips. Each strip represents one of the lines and always faces the viewer (similar to billboards). Strips are textured so that the center is black, representing the line, and the perimeter is white to create the halo. In addition, each strip is bent away from the viewer as shown in Fig. 3(a). This way the part of the strip that is not black prohibits parts of other lines to be drawn that are close in image space but further away from the viewer in depth. This approach is related to the ε-z-buffering used in point-based rendering [3, 11] that uses fragment-dependent depth corrections to determine the visibility of splats in a two-pass process. Our approach, however, makes use of fragment depth manipulations in a single rendering pass to directly render lines that are close together without overlapping halos, similar to Tarini et al.'s depth-aware contour lines [33].

In practice, our approach for rendering 3D lines is a two-stage process. In the first stage we transform the lines into view-oriented triangle strips, while in the second stage we manipulate the shape of the strip and texture it. These two stages are mapped to the two stages in modern GPU processing: vertex shading and fragment shading.

3.2 View-Oriented Triangle Strips

Our goal is to display the line data as view-aligned strips that represent both the line itself as well as the halo around it. Therefore, before sending data to the GPU, we organize our input lines in the CPU as sequences of 3D vertices. To be able to later render the lines as triangle strips on the GPU, we create zero-width line strips on the CPU by duplicating each vertex but retaining the vertex locations. On the GPU, this strip needs to be widened by extending it perpendicular both to the viewing direction and the line direction so that it is always oriented to face the viewer. We derive the direction of the line locally at each of the vertices by taking the normalized average direction of both line segments adjacent to the vertex (or one segment for start and end of each line). This step occurs before the vertices are duplicated, and the direction is copied to the new vertex when the duplication is carried out. As a final pre-processing step, each of the vertices is assigned texture coordinates (u, v). For this purpose, the u-coordinate is interpolated along the length of the line, while the v-coordinate is set to 1 for the "left" side of the strip and to 0 for its "right" side. As a result, each vertex now has a position, texture coordinates, and a direction. This information in the form of zero-width triangle strips is transferred to the GPU as vertex buffer objects.

During the GPU rendering stage (once for each rendering pass), the strip is widened and view-aligned. For this purpose we extend the zero-width strip into a direction that is perpendicular to both the direction of the line $\mathbf{D}$ and the view direction $\mathbf{V}$. Thus, we compute the cross-product between $\mathbf{V}$ and $\mathbf{D}$, normalize the resulting vector, and move a vertex along this direction if $v = 1$ and in opposite direction if $v = 0$. Hence, the new vertex position p_{out} is calculated as:

$$p_{\text{out}} = p_{\text{in}} + \|\mathbf{V} \times \mathbf{D}\|(v - 0.5)w_{\text{strip}}, \tag{1}$$

where p_{in} is the input vertex position and w_{strip} is the strip width. The result is that the strip always faces the viewer, and the centerline of the strip is located along the original line.

3.3 Fragment Texturing and Depth Displacement

The next step in the process is to assign either black or white to the individual pixels of the line strip so that both line and halo are created, but without the halo obstructing nearby lines. This occurs in the fragment shader after the previously created line strip has been rasterized. We first determine the distance (s) to the center of the strip for each fragment, again using the texture coordinates:

$$s = w_{\text{strip}}|v - 0.5|. \tag{2}$$

If this distance s is smaller than half the line width (w_{line}), the fragment's output color is set to black and its depth value is left untouched. Otherwise, the fragment's output color is set to

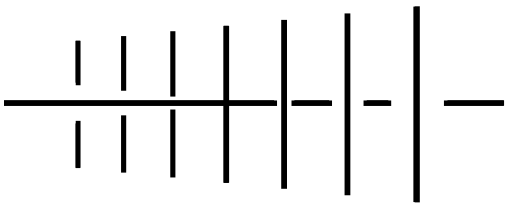


Fig. 4. Illustration of how the line halos change depending on the distance of the lines with respect to each other.

white and its depth is adjusted depending on the distance from the strip's center (Fig. 3(a)):

$$d_{\text{new}} = d_{\text{old}} + d_{\max} f_{\text{displacement}}(2|v - 0.5|). \quad (3)$$

Here, d_{new} is the fragment's new depth, d_{old} is the its old depth, $d_{\max}$ is the maximum displacement, and $f_{\text{displacement}}$ is a function that maps a scalar value $x \in [0,1]$ to $[0,1]$, representing the specific shape of depth displacement. A simple linear function has proven to be suitable, and we use $f_{\text{displacement}}(x) = x$ for all our examples (except Fig. 15).

The effect of this depth displacement of fragments is illustrated in Fig. 3(b) where one line (black) is rendered on top of another one (red). Because depth testing is enabled, parts of the red crossing line are obscured by white fragments of the triangle strip belonging to the black foreground line. The visual effect is a halo around the black foreground line.

If the red line in Fig. 3(b) were to move further back, the width of the halo would increase until the difference in depth between both lines is larger than $d_{\max}$, after which the halo width remains constant. Moving the background line toward the foreground line, in contrast, would decrease the halo width until the halo completely disappears. This happens if the background line is closer to the foreground line than $f_{\text{displacement}}(0.5 w_{\text{line}})$. This effect is illustrated in Fig. 4 in which a series of vertical lines of decreasing distance to the viewer are rendered with respect to a horizontal line. Notice how the changing halo widths enhance the depth perception in this case.

3.4 Visual Enhancements

So far the technique correctly represents haloed lines that do not occlude each other if they lie close together. For an individual line that lies clearly in front of a bundle of other lines this has the effect of showing the rectangular strip—visible, in particular, at the line ends (Fig. 5(a))—which can be distracting. We improve the visual appearance of the lines by tapering (gradually narrowing line ends, Fig. 5(b)). We place a stencil texture over the strip to be checked in the fragment shader which only lets fragments pass that are inside the mask. To make the amount of tapering independent from the line length, the u texture coordinate is, in fact, interpolated non-linearly along each triangle strip. For the first n_{tapered} vertices the u-coordinate is linearly interpolated between 0 and t_{tapered}, assuming they are approximately equidistant. Similarly, the last n_{tapered} vertices are mapped to u-values between $1 - t_{\text{tapered}}$ and 1. We use $n_{\text{tapered}} = 2$ and $t_{\text{tapered}} = 0.2$.

One important aspect of 3D data visualization is to correctly display spatial relationships. While the depth-dependent halo rendering already supports depth perception (Fig. 6(a)), we further enhance this effect using depth cueing by adjusting the line

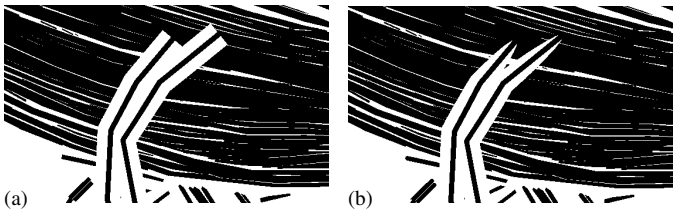


Fig. 5. Comparing rendering (a) without and (b) with tapering.

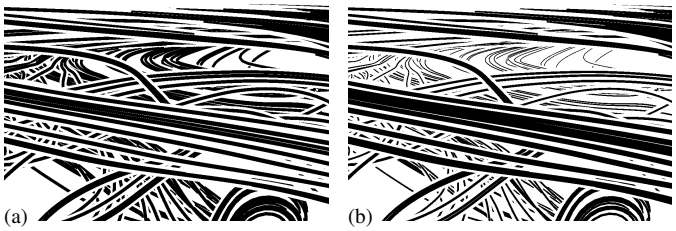


Fig. 6. Rendering (a) without and (b) with additional depth cueing.

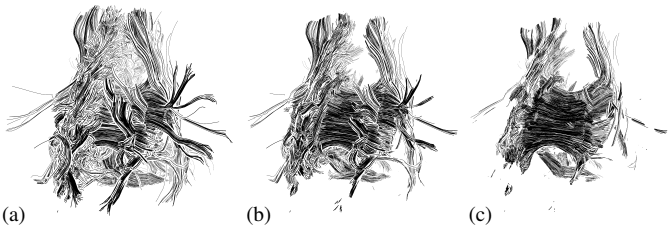
width [10]. The perspective projection that we use already introduces some foreshortening of the line strips with growing distance from the viewer. We further emphasize this effect by manipulating the portion of the strip that is rendered as *line*, reducing it the further away a line section lies from the viewer (Fig. 6(b)). This is realized in the fragment shader by modifying w_{line} with respect to the current fragment's z-value. The degree of depth cueing can be adjusted w.r.t. the desired effect.

Finally, the visual quality of a line rendering depends to a large degree on the way the resulting image is represented. While using vector graphics could be advisable for the line renderings we are producing, we can also achieve similar results by rendering high-resolution 1-bit black-and-white images as used throughout the paper. For on-screen rendering we either use a gradual transition from black to white in the fragment shader or employ full-scene anti-aliasing (FSAA). For example, Coverage Sampling Anti-Aliasing (CSAA) can be enabled to produce higher-quality images with less sampling artifacts.

3.5 Filtering

As another means of illustrative abstraction (in addition to selecting a subset of lines through a ROI) we filter the data to remove selected parts, e. g., low velocity streamlines or fiber tracts in areas with low fractional anisotropy (FA). For this purpose we assign an extra scalar attribute to each vertex of the data based on which the filtering will occur. This filtering attribute is passed onto the GPU and replicated for each fragment in the rasterization stage. The fragment shader then compares each fragment's filtering attribute with a pre-determined threshold and discards fragments that do not pass this test. The threshold can now be changed at run-time and permits an interactive selection of the amount of filtering that is to occur.

Unfortunately, this process re-introduces the problem of the rectangular shape of line ends discussed in Section 3.4. To address it, we change the process of halo masking when filtering is enabled. Instead of using the actual interpolated u texture coordinate we derive a new u-value from the interpolated filtering attribute and the currently active threshold such that the ends of the filtered lines are tapered. Limitations are that this assumes the filter attribute changes gradually and that it does not result



(a) (b) (c)

Fig. 7. Three example stages of filtering, using the fractional anisotropy (FA) value of a DTI fiber tract dataset. With the growing filtering threshold for FA, more of the internal structure of the dataset is revealed.

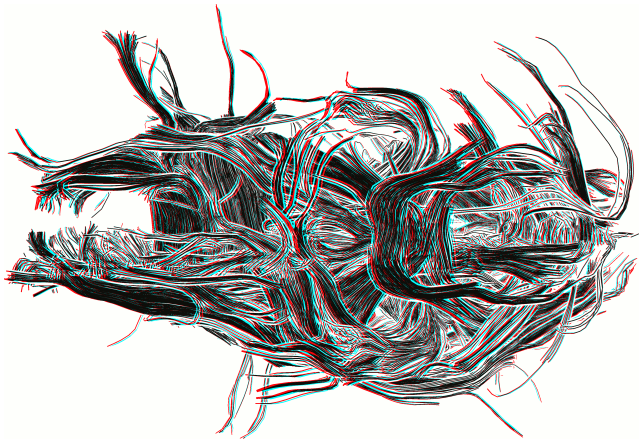


Fig. 8. Anaglyphic stereo visualization of DTI fiber tracts. For use with red-cyan or red-green glasses (red on the left eye).

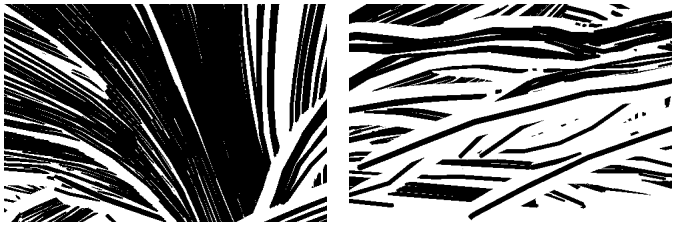
in the same visual quality as before.

The scalar attribute used for filtering needs to be meaningful with respect to the specific dataset. For example, for DTI fiber tract data we use fractional anisotropy, which is a measure for the amount of directionality in the data (Fig. 7). Since this value is defined based on a volume representation, it is interpolated for each vertex in the pre-processing stage based on the vertex' location in the volume.

3.6 Anaglyphic 3D Rendering

The three-dimensionality of the data is one important aspect that needs to be visualized, even beyond the support of halos. In regular rendering for visualization this is achieved through regular shading, potentially combined with special techniques such as depth buffer unsharp masking [20] or ambient occlusion [33]. Alternatively, stereo rendering and projection can be used, i. e., computing separate images for each of the viewer's eyes. The black-and-white character of our visualization technique does not permit using shading-based techniques, but lends itself to stereo rendering without requiring complicated projection setups: anaglyphic rendering. For this purpose we render the scene from two different viewpoints, color these red and cyan, and overlay them on top of each other for use with red-cyan glasses (Fig. 8).

The illustrative line rendering technique lends itself, in particular, to this stereo vision technique because it is monochrome and the discrete elements (lines and points) allow the human visual system to make an easy association between related elements. The halos around lines and line bundles enhance this effect because it makes the separation of individual elements that belong to each other easier. In addition to anaglyphic rendering, we also applied the technique to passive stereo rendering using polarized light, resulting in a comparable experience but without the small color artifacts caused by the red-cyan glasses.



(a) Emphasis of dense bundles. (b) Abstraction for non-aligned lines.

Fig. 9. Illustration principles at work. Detail regions from Fig. 1.

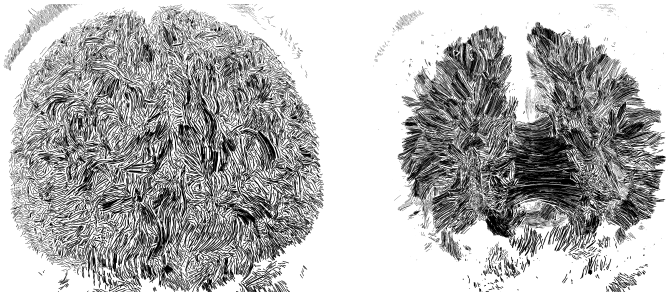
4 Discussion

After having described the illustrative line rendering technique and its implementation in detail, we now discuss some application aspects. In particular, we address how depth-attenuated halos incorporate the illustration principles of abstraction and emphasis, how specific visual results can be achieved, and how the technique can be applied to a number of application scenarios and input data types.

4.1 Illustration Principles in Depth-Attenuated Halos

Our illustrative rendering technique for lines has the effect that no or only small halos are created between lines that form concentrated bundles (Fig. 9(a)). This effect emphasizes line bundles visually, through larger dark regions or tightly packed co-linear lines and through separation from the background. This emphasis may make the individual lines less distinguishable, but supports and highlights the importance and the coherency of such dense line bundles. Lines that do not form concentrated bundles (which are less aligned), in contrast, are visually de-emphasized and abstracted (Fig. 9(b)): For crossing lines that are not at the same distance from the viewer many halos are generated, splitting these lines up into smaller segments. These emphasis and abstraction effects are enhanced further through the usage of additional depth cueing via line width attenuation, emphasizing front parts and de-emphasizing distant parts of the scene. These methods allow us to refrain from using any lighting while still illustrating spatial relations.

The illustrative line rendering technique using depth-attenuated halos has the additional effect that lines behind tight bundles are visually separated due to the halo that is rendered around the bundle. Thus, the technique renders the line data as implicit layers that are visually separated from each other through their halos. Occlusion situations are clearly visible through the halos surrounding the front line bundles, leading to a *visual depth clustering* similar to the halos in [33]. Moreover, this effect also means that for bundles or other co-linear lines only the front-most layer of lines are displayed in the resulting images—visible, in particular, in videos, during interactive manipulation, or when viewing anaglyphic stereo images (Fig. 8). Therefore, despite the fact that no surfaces exist explicitly in the data, the rendering has the interesting effect that surfaces that implicitly exist in the data by means of densely packed bundles are visually noticeable.



(a) Full dataset. (b) Filtered dataset.

Fig. 10. Fiber tracts from diffusion tensor imaging (DTI). Notice the visual bundling and depth clustering in (b).

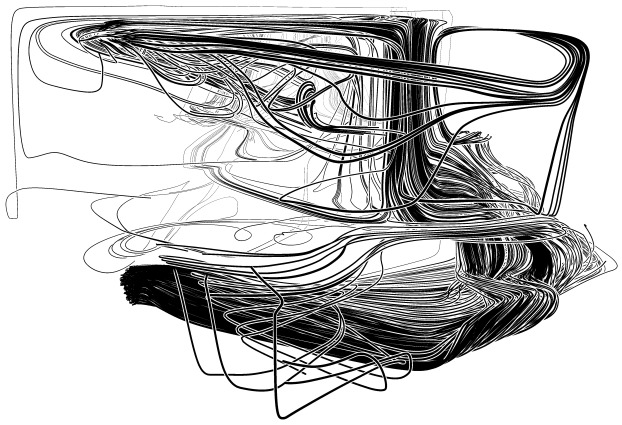


Fig. 11. Simulated air flow in an office.

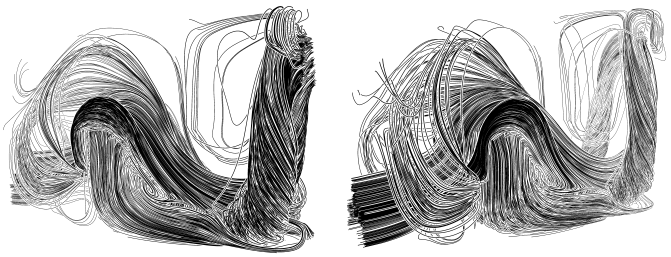
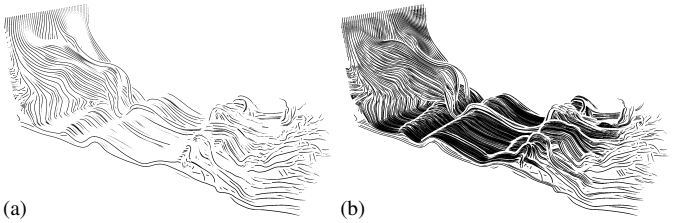


Fig. 12. Two views of a selection of simulated water flow streamlines.



(a) (b)

Fig. 13. Illustrative rendering of streamlines in a heat-driven cavity visualized with a low (a) and a higher (b) value for $d_{\max}$, resulting in two different ways of depicting the sheets.

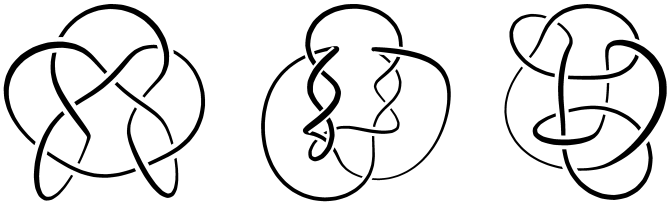


Fig. 14. Visualizations of simple mathematical shapes from knot theory.

4.2 Case Studies of Application Scenarios

To illustrate the applicability of our technique we now discuss a number of case studies from a variety of domains.

4.2.1 DTI Fiber Tracts

Nerve fiber tracts extracted from diffusion tensor imaging (DTI) give an indication of how actual bundles of axons connect different parts of the brain. Depending on the resolution of the underlying MRI scan, a large number of fibers can be extracted. In our example in Fig. 10, 150 352 tracts with 1 625 472 vertices in total were extracted using the program Diffusion Toolkit, the small average vertex number per tract resulting from many short tracts. Because of such large dataset sizes, fibers are difficult to visualize with traditional techniques due to performance, rendering technique (shaded cylinders need a certain amount of space), and overview/occlusion issues. Hence, usually subsets of the fiber tracts are selected and visualized. Our illustrative line rendering technique can easily render whole datasets at interactive rates but also suffers from overview/occlusion issues (Fig. 10(a)). Thus, we also select subsets (e. g., Fig. 1 and 8, where the ROI is a sagittal slice) and/or use filtering to cope with this type of data (e. g., Fig. 7 and 10(b)). Nevertheless, our technique is able to display all fibers in a subset so that no automated data reduction technique is necessary that would cluster several lines into single ones. With our technique it is possible to *visually* distinguish single fibers from smaller or larger fiber bundles rather than requiring algorithmic support for this task.

4.2.2 Simulated Flows of Fluids or Gases

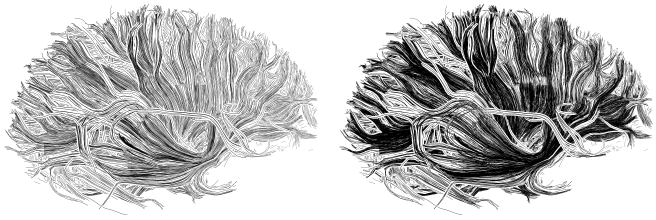
Another domain where line data is generated is the simulation of fluids or gases. For example, we used VTK's "office" example dataset and extracted 786 streamlines from it using VTK (Fig. 11). The visualization in Fig. 11 shows where the simulated air flow is focused and where it branches off, highlighting concentrated bundles of air movement. Fig. 12 shows two views of a similar simulation of water flow using 1 400 streamlines and 2 603 605 vertices in total. Here, the twisted flow of the water in a number of vortices is illustrated, while focused flow is still emphasized. The combination of these illustrative effects gives the illustrations a vividly three-dimensional appearance. Fig. 13 shows streamlines in a heat-driven cavity. These visualizations show that some groups of streamlines visually form sheet-like structures and that the depiction of 'sheetness' can be influenced by choosing d_{max} smaller (Fig. 13(a)) or larger (Fig. 13(b)). The presence of these sheets of lines makes this dataset of lines ideal for our technique.

4.2.3 Mathematical Shapes

As a last example, Fig. 14 shows a selection of much simpler and less dense datasets from knot theory. This illustrates that our method can also handle simple shapes with similar results as in previous work [10].

4.3 Parametrization

A good set of parameters depends to a certain degree on the specific dataset. For the illustrations in this paper we chose



(a) $f_{\text{displacement}}(x) = x^2$ (b) $f_{\text{displacement}}(x) = \sqrt{x}$

Fig. 15. The effect of using nonlinear displacement functions.

Table 1. Performance measurements, sorted by dataset size.

Fig.	Lines	Vertices	Anagl.	Frame rate
1	11 306	260 836	no	123fps
8	11 306	260 836	yes	65fps
11	786	278 849	no	290fps
10(a)	150 352	1 625 472	no	24fps
12	1 400	2 603 605	no	43fps

w_{line} and w_{strip} such that w_{line} is 4–8 times as large as w_{strip}, for point data the factor was 8–27. The size of w_{line} also depends on the specific data and needs to be smaller for datasets with more elements. The maximum displacement, $d_{\max}$, is set to 0.01 for all illustrations in this paper. Setting $d_{\max}$ to zero has the effect of giving all lines a halo, potentially blocking other lines, see Fig. 2(c). In general, once a good setup for these parameters is found for a given dataset, they do not need to be changed for interactive exploration and filtering or when visualizing lines resulting from a different ROI.

Unless otherwise indicated, we always use the same linear displacement function $f_{\text{displacement}}(x) = x$. Changing this function can easily be accomplished by adapting the function in the shader or by using a non-linear gray-ramp texture. The resulting visual effect differs to some degree from previously shown images as shown in Fig. 15 and can be used to achieve the impression of more or less densely packed bundles. For example, using $f_{\text{displacement}}(x) = x^2$ results in less emphasis for the bundles but more individual lines being visible (Fig. 15(a)), while $f_{\text{displacement}}(x) = \sqrt{x}$ has the opposite effect (Fig. 15(b)).

4.4 Performance and Limitations of the Technique

In Table 1 we give average frame rates for a selection of datasets. The measurements were taken on a 3GHz Intel Core2 Extreme with 4GB RAM, running Windows Vista, and using an NVIDIA GeForce 8800 GTX graphics card. The rendering performance for both lines and points is determined mainly by the number of lines, the number of vertices, the size of the dataset on the screen, and the size of the strips or quads. For the measurements, we maximized each visualization on the 812×600 pixel rendering window and then measured the average speed for animating a rotation, with vertical sync disabled and without anti-aliasing or filtering. The offline pre-processing to extract streamlines from 3D vector data took in the order of several seconds to several minutes and was done with VTK.

Besides the obvious limit in the number of lines and vertices that the technique can handle as just indicated by the performance data, there are two aspects that are of further importance. One is that, while the technique works well when the data elements inherently form a surface or closely bundled structures, it performs not as well for datasets with less structured but still dense elements. For example, if a volume were to be filled with a dense set of random lines, the technique would not create meaningful visualizations. The second limitation is that the technique performs best for visualizing the outer layer of closely bundled elements. While filtering does allow to look at subsets of the data, it still does not allow to look beyond the surface of the visible structures. Transparency based on a given parameter cannot easily be added because this would require sorting based on individual line segments which would greatly diminish the technique's performance and, thus, its suitability to large and dense line datasets.

5 Conclusion

We have presented a technique for illustrative visualization of dense line datasets. We create depth-attenuated halos around lines that do not overlap each other if the lines are close in depth, but do occlude lines located further away in depth. This has the effect of emphasizing tightly bundled line structures and abstracting from less organized regions in the data. An additional important illustrative effect is the resulting visual depth clustering of visually connected regions. The visually connected regions portray surfaces implicitly existing in the dataset due to its inherent structure. These illustrative rendering techniques are augmented by traditional interaction and line rendering techniques such as filtering and depth cueing. While supporting the rendering at interactive to real-time frame rates depending on the size of the dataset, the technique is also capable of producing high-quality black-and-white renderings so that the results can easily be used in printed materials.

An informal evaluation with domain experts revealed that our visualizations are successful in illustrating brain fiber structures, showing detail, emphasizing fiber bundles, and depicting spatial relationships. While these results are encouraging, we need to continue our evaluation and explore both a richer collection of data sources (i. e., other than DTI) and evaluate the technique in the other domains for which we have created visualizations. In particular, one of the requests in the informal evaluation was to provide additional context for the line rendering, which was recently realized by Svetachov et al. [32].

Additional future work includes, for example, adapting the process of turning lines into view-oriented triangle strips which consists of a data duplication part that unnecessarily uses too much memory. This can be avoided by using modern graphics cards extensions such as geometry shaders or the instanced arrays extension. The visual appearance could also be improved by exploring alpha-blending, for instance guided by the filtering process. By employing line style rendering techniques such as used in [38] it may even be possible to maintain the black-and-white character necessary for high-quality print output.

Acknowledgments

We thank Cris Lanting and Pim van Dijk from the BCN NeuroImaging Center in Groningen, NL, for the brain datasets used in this paper. The turbulent flow simulation dataset is courtesy of Martin Rumpf, Univ. Bonn, Germany. The heat driven cavity dataset is courtesy of Roel Verstappen, University of Gronin-

gen. We also thank Alessandro Crippa, Moritz Gerl, Wladimir van der Laan, and Alex Telea for interesting discussions. This research is funded by the Dutch National Science Foundation (NWO), "VIEW" program, project no. 643.100.501.

REFERENCES

[1] A. Appel. The Notion of Quantitative Invisibility and the Machine Rendering of Solids. In *Proc. 22nd ACM National Conference*, pp. 387–393, New York, 1967. ACM. DOI: 10.1145/800196.806007

[2] A. Appel, F. J. Rohlf, and A. J. Stein. The Haloed Line Effect for Hidden Line Elimination. *ACM SIGGRAPH Computer Graphics*, 13(3):151–157, Aug. 1979. DOI: 10.1145/800249.807437

[3] M. Botsch and L. Kobbelt. High-Quality Point-Based Rendering on Modern GPUs. In *Proc. Pacific Graphics*, pp. 335–343, Los Alamitos, 2003. IEEE Computer Society. DOI: 10.1109/PCCGA.2003.1238275

[4] S. Bruckner and E. Gröller. Enhancing Depth-Perception with Flexible Volumetric Halos. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1344–1351, 2007. DOI: 10.1109/TVCG.2007.70555

[5] M. Burns, J. Klawe, S. Rusinkiewicz, A. Finkelstein, and D. DeCarlo. Line Drawings from Volume Data. *ACT Transactions on Graphics*, 24(3):512–518, July 2005. DOI: 10.1145/1073204.1073222

[6] B. Cabral and L. C. Leedom. Imaging Vector Fields using Line Integral Convolution. In *Proc. SIGGRAPH*, pp. 263–270, New York, 1993. ACM. DOI: 10.1145/166117.166151

[7] O. Deussen and T. Strothotte. Computer-Generated Pen-and-Ink Illustration of Trees. In *Proc. SIGGRAPH*, pp. 13–18, New York, 2000. ACM. DOI: 10.1145/344779.344792

[8] F. Dong, G. J. Clapworthy, H. Lin, and M. A. Krokos. Nonphotorealistic Rendering of Medical Volume Data. *IEEE Computer Graphics & Applications*, 23(4):44–52, July/Aug. 2003. DOI: 10.1109/MCG.2003.1210864

[9] D. Ebert and P. Rheingans. Volume Illustration: Non-Photorealistic Rendering of Volume Models. In *Proc. Visualization*, pp. 195–202, Los Alamitos, 2000. IEEE Computer Society. DOI: 10.1109/VISUAL.2000.885694

[10] G. Elber. Line Illustrations ∈ Computer Graphics. *The Visual Computer*, 11(6):290–296, June 1995. DOI: 10.1007/s003710050022

[11] M. Gross and H. Pfister, editors. *Point-Based Graphics*. Elsevier, 2007.

[12] H. Hauser, L. Mroz, G. I. Bischi, and M. E. Gröller. Two-Level Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252, July–Sept. 2001. DOI: 10.1109/2945.942692

[13] A. Helgeland and O. Andreassen. Visualization of Vector Fields Using Seed LIC and Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):673–682, Nov./Dec. 2004. DOI: 10.1109/TVCG.2004.49

[14] A. Hertzmann and D. Zorin. Illustrating Smooth Surfaces. In *Proc. SIGGRAPH*, pp. 517–526, New York, 2000. ACM. DOI: 10.1145/344779.345074

[15] V. Interrante and C. Grosch. Visualizing 3D Flow. *IEEE Computer Graphics & Applications*, 18(4):49–53, July 1998. DOI: 10.1109/38.689664

[16] A. Joshi, J. Caban, P. Rheingans, and L. Sparling. Case Study on Visualizing Hurricanes Using Illustration-Inspired Techniques. *IEEE Transactions on Visualization and Computer Graphics*, 2009. To appear. DOI: 10.1109/TVCG.2008.105

[17] M. Kaplan. Hybrid Quantitative Invisibility. In *Proc. NPAR*, pp. 51–52, New York, 2007. ACM. DOI: 10.1145/1274871.1274879

[18] J. Klein, F. Ritter, H. K. Hahn, J. Rexilius, and H.-O. Peitgen. Brain Structure Visualization using Spectral Fiber Clustering. In *Research Posters of SIGGRAPH*, article no. 168, New York, 2006. ACM. DOI: 10.1145/1179622.1179816

[19] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum*, 23(2):203–221, June 2004. DOI: 10.1111/j.1467-8659.2004.00753.x

[20] T. Luft, C. Colditz, and O. Deussen. Image Enhancement by Unsharp Masking the Depth Buffer. *ACM Transactions on Graphics*, 25(3):1206–1213, July 2006. DOI: 10.1145/1141911.1142016

[21] K.-L. Ma, G. Schussman, B. Wilson, K. Ko, J. Qiang, and R. Ryne. Advanced Visualization Technology for Terascale Particle Accelerator Simulations. In *Proc. Supercomputing*, pp. 19–30, Los Alamitos, 2002. IEEE Computer Society. DOI: 10.1109/SC.2002.10007

[22] 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, Sept./Oct. 2006. DOI: 10.1109/TVCG.2006.197

[23] S. Mori and P. C. van Zijl. Fiber Tracking: Principles and Strategies – A Technical Review. *NMR Biomed*, 15(7-8):468–480, Nov./Dec. 2002. DOI: 10.1002/nbm.781

[24] Z. Nagy, J. Schneider, and R. Westermann. Interactive Volume Illustration. In B. Girod, H. Niemann, H.-P. Seidel, G. Greiner, and T. Ertl, editors, *Proc. Vision, Modeling and Visualization*, pp. 497–504, Berlin, 2002. Akademische Verlagsgesellschaft Aka GmbH.

[25] V. Petrovic, J. Fallon, and F. Kuester. Visualizing Whole-Brain DTI Tractography with GPU-based Tuboids and LoD Management. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1488–1495, Nov./Dec. 2007.

[26] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. Feature Extraction and Visualization of Flow Fields. In *Eurographics 2002 State of the Art Reports*, pp. 69–100. Eurographics Assoc., Aire-la-Ville, Switzerland, 2002.

[27] F. Ritter, C. Hansen, V. Dicken, O. Konrad, B. Preim, and H.-O. Peitgen. Real-Time Illustration of Vascular Structures. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):877–884, Sept./Oct. 2006. DOI: 10.1109/TVCG.2006.172

[28] G. Schussman and K.-L. Ma. Scalable Self-Orienting Surfaces: A Compact, Texture-Enhanced Representation for Interactive Visualization of 3D Vector Fields. In *Proc. Pacific Graphics*, pp. 356–365, Los Alamitos, 2002. IEEE Computer Society. DOI: 10.1109/PCCGA.2002.1167879

[29] C. Stoll, S. Gumhold, and H.-P. Seidel. Visualization With Stylized Line Primitives. In *Proc. Visualization*, pp. 695–702, Los Alamitos, 2005. IEEE Computer Society. DOI: 10.1109/VIS.2005.124

[30] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey. How to Render Frames and Influence People. *Computer Graphics Forum*, 13(3):455–466, Aug. 1994. DOI: 10.1111/1467-8659.1330455

[31] N. A. Svakhine, Y. Jang, D. S. Ebert, and K. Gaither. Illustration and Photography Inspired Visualization of Flows and Volumes. In *Proc. Visualization*, pp. 687–694, Los Alamitos, 2005. IEEE Computer Society. DOI: 10.1109/VIS.2005.53

[32] P. Svetachov, M. H. Everts, and T. Isenberg. DTI in Context: Illustrating Brain Fiber Tracts In Situ. *Computer Graphics Forum*, 29(3):1024–1032, June 2010. DOI: 10.1111/j.1467-8659.2009.01692.x

[33] M. Tarini, P. Cignoni, and C. Montani. Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1237–1244, Sept./Oct. 2006. DOI: 10.1109/TVCG.2006.115

[34] S. M. F. Treavett and M. Chen. Pen-and-Ink Rendering in Volume Visualisation. In *Proc. Visualization*, pp. 203–210, Los Alamitos, 2000. IEEE Computer Society. DOI: 10.1109/VISUAL.2000.885696

[35] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-Driven Volume Rendering. In *Proc. Visualization*, pp. 139–145, Los Alamitos, 2004. IEEE Computer Society. DOI: 10.1109/VISUAL.2004.48

[36] A. Wenger, D. F. Keefe, S. Zhang, and D. H. Laidlaw. Interactive Volume Rendering of Thin Thread Structures within Multivalued Scientific Data Sets. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):664–672, Nov./Dec. 2004. DOI: 10.1109/TVCG.2004.46

[37] G. A. Winkenbach and D. H. Salesin. Computer-Generated Pen-and-Ink Illustration. In *Proc. SIGGRAPH*, pp. 91–100, New York, 1994. ACM. DOI: 10.1145/192161.192184

[38] J. Zander, T. Isenberg, S. Schlechtweg, and T. Strothotte. High Quality Hatching. *Computer Graphics Forum*, 23(3):421–430, Sept. 2004. DOI: 10.1111/j.1467-8659.2004.00773.x

[39] L. Zhukov and A. H. Barr. Oriented Tensor Reconstruction. In C. D. Hansen and C. R. Johnson, editors, *The Visualization Handbook*, chapter 15, pp. 313–326. Elsevier, Oxford, UK, 2004.

[40] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive Visualization of 3D-Vector Fields Using Illuminated Stream Lines. In *Proc. VIS*, pp. 107–113, Los Alamitos, 1996. IEEE Computer Society. DOI: 10.1109/VISUAL.1996.567777