Opacity Optimization for 3D Line Fields

Tobias Günther Christian Rössl Holger Theisel
University of Magdeburg

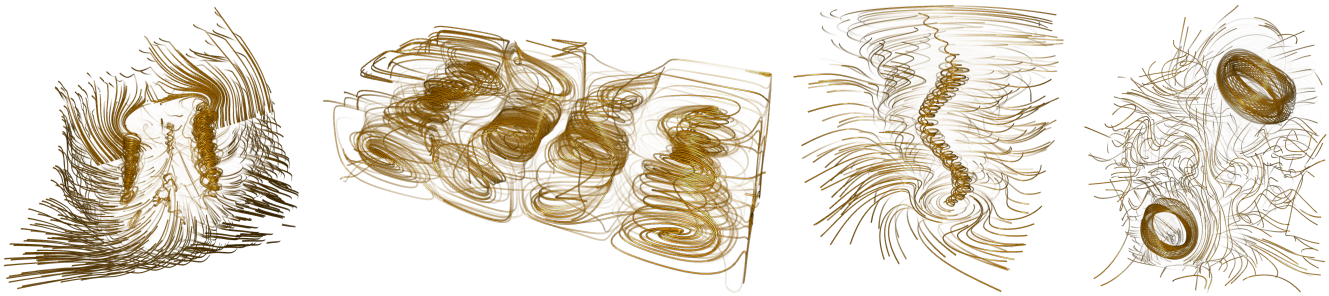


Figure 1: *Applications of our interactive, global line selection algorithm. Our bounded linear optimization for the opacities reveals user-defined important features, e.g., vortices in rotorcraft flow data, convection cells in heating processes (Rayleigh-Bénard cells), the vortex core of a tornado and field lines of decaying magnetic knots (from left to right).*

Abstract

For the visualization of dense line fields, the careful selection of lines to be rendered is a vital aspect. In this paper, we present a global line selection approach that is based on an optimization process. Starting with an initial set of lines that covers the domain, all lines are rendered with a varying opacity, which is subject to the minimization of a bounded-variable least-squares problem. The optimization strives to keep a balance between information presentation and occlusion avoidance. This way, we obtain view-dependent opacities of the line segments, allowing a real-time free navigation while minimizing the danger of missing important structures in the visualization. We compare our technique with existing local and greedy approaches and apply it to data sets in flow visualization, medical imaging, physics, and computer graphics.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms

Keywords: scientific visualization, flow visualization, line fields

Links: DL PDF

1 Introduction

Line fields consist of families of 3D curves that cover (part of) a 3D domain *densely*. They have many applications in scientific visualization (e.g., streamlines and pathlines of velocity vector fields), medical imaging (e.g., tensor lines or fiber bundles of DT-MRI data), physics (e.g., magnetic field lines) and computer graphics (e.g., speed lines to depict motion). With the ongoing development of graphics hardware, anti-aliased, high-quality rendering of massive sets of line primitives has become generally available. However, the main challenge in rendering line fields is *line selection*: from the potentially infinite set of possible lines, a set of representatives has to be selected for rendering, and this selection should visually convey the main features of the data. On the one hand, displaying too many lines results in cluttered renderings where important features may be hidden. On the other hand, displaying too few or the wrong lines may also lead to missing features due to undersampling of the interesting regions.

Line selection for line fields was intensively studied, mainly in the field of flow visualization. So far, all existing methods use a local or greedy approach: a suitable line is found either by locally searching for a good seeding point for a line integration, by a greedy algorithm of repeatedly inserting new lines, or by computing local importance measures for a finite set of pre-selected lines. Furthermore, none of the existing approaches is readily applicable to a free navigation in a scene: existing methods depict lines to generate illustrations for a distant viewpoint, and they do not handle the massive occlusion that can be introduced by even a single line very close to the camera. (Lines are usually expanded to ribbons or tubes to provide depth cues, thus they typically cover more screen space when moving close to them).

This paper is based on the insight that line selection should be formulated as a *global optimization problem*: if a line is detected to be important, but at the same time occludes more important structures, it should not be rendered. On the other hand, if a line is of only moderate importance and does not occlude more important structures, it can (and should) safely be rendered. This means that the decision on selecting a particular line is a compromise between having a maximal amount of conveyed information and having a minimal amount of occlusion of other features. Similar to existing methods, our approach starts out with a finite set of initial lines that cover a 3D domain densely. Instead of selecting a subset of these lines for rendering, we render all lines but assign varying opacities to the line segments. The opacities of segments are repeatedly computed as the minimizers of a quadratic error function, which is possible at interactive rates. This way, we resolve occlusions by locally fading out line segments and attain frame coherence. With this, we introduce the first method that allows for a free, interactive navigation in a scene, while achieving a view-dependent, globally optimal selection of lines from a precomputed set of candidates. In

ACM Reference Format
Günther, T., Rössl, C., Theisel, H. 2013. Opacity Optimization for 3D Line Fields.
ACM Trans. Graph. 32, 4, Article 120 (July 2013), 8 pages. DOI = 10.1145/2461912.2461930
http://doi.acm.org/10.1145/2461912.2461930.

DOI: http://doi.acm.org/10.1145/2461912.2461930

addition, the user can optionally define an importance measure, i.e., incorporate available domain knowledge to emphasize features.

2 Related Work

Early work on line selection was done for streamlines in 2D vector fields [Turk and Banks 1996; Jobard and Lefer 1997; Verma et al. 2000; Jobard and Lefer 2001; Mebarki et al. 2005; Liu et al. 2006; Li et al. 2008]. The focus of these approaches was mainly on finding seeds to cover the domain densely with streamlines. Extensions to 3D have been developed by guiding the seeding by density-based [Mattausch et al. 2003], feature-based [Ye et al. 2005; Yu et al. 2012], or similarity-based measures [Chen et al. 2007; McLoughlin et al. 2012].

In 3D, however, the maximization of perceivable information is not only a matter of evenly sampling the domain, but also of avoiding occlusions: it becomes a view-dependent problem. Li and Shen [2007] applied the iterative 2D seeding strategy of Jobard and Lefer [1997] to produce evenly-spaced streamlines by an image-space approach. Thereby, the depth of the lines is acquired by reprojection. Annen et al. [2008] proposed a seeding algorithm, inspired by non-photorealistic rendering techniques. Hereby, the termination of streamline integration depends on local measures, such that selected lines have the most similar behavior to contours on surfaces. Xu et al. [2010] used an information-theoretic approach to measure the information conveyed by a given set of streamlines. The disparity to the original field's entropy guides the seeding of streamlines to add details where needed. Lines are faded out to reduce occlusion by mapping the scalar entropy field to transparency.

Selection-based approaches do not search for seed points for line integration but instead select lines from a precomputed set. This does not only speed up the search, it also makes frame coherence (i.e., avoidance of popping artifacts) easier to achieve and is therefore a good choice when aiming at interactive navigation. Marchesin et al. [2010] joined lines from a precomputed set with additionally generated lines, depending on the on-screen footprint of the already chosen lines (accumulated in a so-called occupancy buffer), as well as local properties per line. Such local properties again originate in information theory and are the linear entropy [Furuya and Itoh 2008] (i.e., variation of velocity along a line) and the angular entropy [Marchesin et al. 2010] (i.e., variation of a line's direction). To this end, these approaches are neither interactive nor frame coherent. Another local, but view-dependent measure was introduced by Günther et al. [2011], who mapped the number of visible pixels to transparency to fade out lines with only minor contribution. Since this approach tends to favor lines closer to the camera it cannot remove lines covering up the viewport. Ma et al. [2013] combined lines from a view-independent and view-dependent set by considering coherence between local views and the last frame. For filling the viewport they compute occupancy [Marchesin et al. 2010], thus do not account for the order of occlusions.

A problem related to line selection – and sometimes solved simultaneously with it – is viewpoint selection. Lee et al. [2011] steered the selection of lines from a precomputed pool in a greedy algorithm based on the maximum intensity projection (MIP) of a scalar entropy field, called maximum entropy projection (MEP). They also used the MEP to select the best viewpoint among 780 candidates placed on a sphere surrounding the data set. Tao et al [2013] coupled the selection of streamlines and viewpoints by modeling them as interrelated information channels [Wang and Shen 2011], i.e., they selected a streamline set and a viewpoint simultaneously. The viewpoint is constrained to be located on a sphere surrounding the data set, and the streamline selection is invariant under camera movement but is based on the evaluation of multiple views. Additionally, they generate a camera path on the sphere.

All existing approaches for 3D line selection are – to the best of our knowledge – either local or greedy. "Local" means that a line is selected based on certain importance measures without considering its relation to other selected lines. "Greedy" means that a new line is selected based on its relation to already selected lines, making the process of line selection dependent on the order of line insertion. Furthermore, the restriction of some techniques for having the viewpoint outside of the domain is a limitation that makes viewpoint selection approaches not an option for free scene navigation.

Table 1 compares features of the most relevant techniques for 3D line selection and our new approach. We consider (in columns left to right) the following features: view dependence (selected lines change when changing the viewpoint), frame coherence (small changes of the viewpoint lead to small changes in the shown lines without popping artifacts of new lines; view-independent techniques are trivially frame coherent), occlusion (occlusion information is used for line selection), interactivity (line selection is either precomputed or fast enough for interactive navigation, trivially yes for view-independent), GPU-accelerated (availability of GPU-accelerated implementation), transparency (semi-transparency used for line rendering), and strategy (either local, greedy, or global). The yellow boxes could not be uniquely classified: the image-based approach of Li and Shen [2007] produces for a given view a line set that does not intersect in image-space, thus there is no occlusion. For more complex and larger data sets they propose to combine the lines acquired from multiple views. In that case, they do not address the arising occlusion. Albeit Li and Shen [2007] and Ma et al. [2013] first validate lines from the previous frame, frame coherence is still an issue, as newly added lines introduce popping

Method	View-depend.	Frame coher.	Occlusion	Interact.	GPU-acc.	Transp.	Strategy
[Chen et al. 2007]	✗	✓	✗	✓	✗	✗	greedy
[Li and Shen 2007]	✓	✓	✓	✗	n/a	✗	greedy
[Annen et al. 2008]	✓	✓	✗	✓	✓	✗	local
[Marchesin et al. 2010]	✓	✗	✓	✗	✓	✗	greedy
[Xu et al. 2010]	✗	✓	✗	✓	n/a	✓	greedy
[Günther et al. 2011]	✓	✓	✗	✓	✓	✓	local
[Lee et al. 2011]	✓	✗	✓	✓	✓	✗	greedy
[McLoughlin et al. 2012]	✗	✓	✓	✓	n/a	✗	greedy
[Yu et al. 2012]	✗	✓	✓	✓	✓	✗	greedy
[Tao et al. 2013]	✗	✓	✗	✓	✓	✗	greedy
[Ma et al. 2013]	✓	✓	✓	✓	✓	✗	greedy
Our approach	✓	✓	✓	✓	✓	✓	**global**

Table 1: *Comparison of features of related methods and our approach.*

artifacts. McLoughlin et al. [2012] and Yu et al. [2012] present hierarchical streamline clustering algorithms in which the user steers the degree of occlusion by interactively selecting a level-of-detail.

3 Problem Setting and Error Function

Our approach starts with a finite set of polylines, which cover the domain densely. Instead of selecting particular polylines or polyline segments, all polylines of the initial set are rendered but with a variable opacity that minimizes an error functional. Every polyline is split into a number of segments, producing in total n polyline segments. Figs. 2(a) and 2(b) illustrate this for two polylines with $n = 6$. We compute the optimal opacity $\alpha_i \in [0, 1]$ for each segment $1 \leq i \leq n$ as solution to a bounded-variable least-squares problem (Fig. 2(c)). For rendering, the opacities are interpolated between adjacent segments to yield vertex opacities, see Fig. 2(d).

Each segment is equipped with a local importance $g_i \in [0, 1]$, which can be chosen depending on the application. (A discussion of possible choices for g_i follows in Section 5.1.) The higher g_i the more the segment should be emphasized. If no particular importance is given, g_i is set to 0.5 for all segments. In addition, the following properties are computed for every pair (i, j) of segments

- a_{ij} encodes adjacency of segments: $a_{ij} = 1$ if the segments i and j are adjacent on the same polyline; otherwise $a_{ij} = 0$. The example in Figure 2(c) has $a_{12} = a_{21} = a_{23} = a_{32} = a_{45} = a_{54} = a_{56} = a_{65} = 1$ and all remaining $a_{ij} = 0$,
- $h_{ij} \in [0, 1]$ describes how much segment j is occluded by segment i as seen from the particular viewpoint. Its continuity during camera movement induces frame coherence. In Figure 2(c) we have $h_{25} > 0$ and all remaining $h_{ij} = 0$.

We find opacities α_i as minimizers of the quadratic error function

$$E = p \sum_{i=1}^{n} (\alpha_i - 1)^2 \tag{1}$$

$$+ q \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\alpha_i \, (1 - g_i)^\lambda \, h_{ij} \, g_j \right)^2 \tag{2}$$

$$+ r \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\alpha_i \, (1 - g_i)^\lambda \, h_{ji} \, g_j \right)^2 \tag{3}$$

$$+ s \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} (\alpha_i - \alpha_j)^2 \tag{4}$$

with *bounded* variables $0 \leq \alpha_i \leq 1$. The four terms of E can be interpreted as follows: (1) is a regularization to prevent almost empty renderings, i.e., prevent all α_i from being close to 0 by penalizing a variation of α_i from 1. (2) introduces a penalty if an important segment j (i.e., g_j is large) is occluded by an unimportant segment i (i.e., $1 - g_i$ is large). Then, $h_{ij} > 0$ and in this case the opacity of segment i should be rather low, i.e., it should not occlude segment j. The parameter λ steers the fall-off of g_i from 1. (3) covers the opposite case, i.e., an unimportant segment i is occluded by an important segment j. Similarly, the opacity of segment i is forced to be small in order to have a "more empty" background behind important segment j. (4) is a smoothness term that enforces a slow change of the opacity along a line. Finally, the weights p, q, r, s, balance the contribution of the terms. They can be restricted by a normalization or global scale, e.g., we choose $p = 1$. Section 5.2 discusses the parameters in more detail.

4 Details and Implementation

The essence of our approach is to use transparency to locally fade out line segments that cause occlusion. This requires a method for correctly blending many transparent layers, see Maule et al. [2011] for a survey of raster-based transparency techniques. We decided to create on the GPU for each pixel a linked list of the rasterized fragments [Yang et al. 2010], as this data structure serves two different purposes: blending of transparent layers and generation of penalties h_{ij} that measure occlusion caused by segment i on j. For the latter, fragment linked list elements store – in addition to color and depth – a value that is used to identify the polyline segments. Based on the segment index and the fragment depth, the order of occlusions is taken into account.

4.1 Initial Line Set

Firstly, we obtain the initial polyline set from the given line field in a preprocess. For this, the only requirement is a dense covering of the domain, i.e., for every point in the domain there is a line passing by closely. For the examples on flow visualization and medical imaging, we used a random distribution of seeding points for line integration. If random seeding does not yield a fair uniform distribution of line geometry, sophisticated view-independent seeding strategies may serve as input [Chen et al. 2007; Xu et al. 2010; McLoughlin et al. 2012]. Secondly, we partition each polyline into a fixed number of k polyline segments. The total number of segments is n. The value k trades the potential variation in opacity along each polyline for the size of the system to solve. We observed that k can be rather small because the computed opacities are blended smoothly along polylines for rendering. We used $k = 8$ for all examples.

4.2 Fragment Linked List Construction and Rendering

All element data of the fragment linked lists resides in a global memory pool that is shared by all pixels, and is managed by atomic operations in the fragment shader. The element data to append is obtained by rendering all lines with depth-dependent halos [Everts et al. 2009], i.e., we expand the lines in a geometry shader to viewport-aligned triangle strips and alter the depth and color of the boundary – halo – region. Other depth enhancing method are imaginable here as well, e.g. line ambient occlusion [Eichelbaum et al. 2013] or unsharp masking of the depth buffer [Luft et al. 2006]. We use an illuminated streamline shading (ISL) by Zöckler et al. [1996], combined with multi-sampling (16× coverage-sampling anti-aliasing/CSAA, i.e., 4 samples per pixel). We run the pixel shader only once per pixel (i.e., on pixel-frequency) to create only one fragment in the linked list, and evaluate multiple samples explicitly to smooth the sharp edge between center and halo region.

(a) Given is a set of polylines. (b) Discretize polylines into n segments (here: $n = 6$). (c) Compute per-segment opacity α_i by energy minimization. (d) Interpolate opacities between adjacent segments for final rendering.

Figure 2: *Illustration of the general idea.*

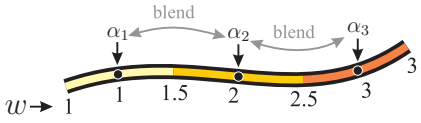


Figure 3: *The precomputed parameterization w provides lookup up of nearest line segment centers and blending weights.*

Every opacity value is associated with the center of the respective polyline segment. The arc-length distance of an arbitrary point on the line to closest left and right centers is used as weight for blending opacities, i.e., to define the transparency of a line's fragment. Blending weights and the indices of the closest opacity centers are easy to obtain from the blending weight parameterization w, shown in Figure 3, which is precomputed and added as a vertex attribute. We evaluate index i and opacity α for a weight w as

$$\begin{aligned} i &\leftarrow \text{floor}(w) \\ \alpha &\leftarrow \text{lerp}\big(\alpha_i, \alpha_{i+1}, \text{frac}(w)\big) \,. \end{aligned}$$

The transparent colored fragment is appended to the pixel's fragment linked list. We store the depth (quantized to 24 bit) and the coverage bit vector (8 bit) together. Color and blending weight w are stored as RGBA vector with 8 bit per component and as full 32-bit float, respectively. A second pass sorts the lists, which is required for computing h_{ij} (see below). We remark that also rendering benefits from pre-sorting per pixel instead of sorting per sample.

4.3 Computation of h_{ij}

To gain higher throughput, we use an otherwise unoccupied CPU core for the assembly of the sparse matrix H. Thus, in order to compute the values h_{ij}, we stream the fragment linked lists asynchronously to the CPU after sorting. We iterate all lists and sum up the weights of the occluded fragments for all segments, see Alg. 1.

```
Input: fragment linked list per pixel
Output: h(i,j)
h(i,j) ← 0 ;
foreach pixel p do
    foreach fragment A in linked list of pixel p do
        i ← floor(A.w + 0.5) ;
        foreach fragment B behind A do
            j ← floor(B.w) ;
            v ← frac(B.w) ;
            h(i,j) ← h(i,j) + 1-v ;
            h(i,j+1) ← h(i,j+1) + v ;
        end
    end
end
```

Algorithm 1: Penalty h_{ij} for segment i occluding j.

4.4 Minimization of the Error Function E

The minimization of the quadratic error function E refers to solving a bounded-variable least-squares problem which can be formulated in the normal equations as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} + \text{const} \\ \text{subject to} \quad & 0 \le \mathbf{x}_i \le 1. \end{aligned}$$

with $\mathbf{x} = (\alpha_1, ..., \alpha_n)^T$. The sparse, positive definite matrix $\mathbf{Q}$ captures the error terms (1)-(4) together with the constant vector $\mathbf{c} = (-1, \ldots, -1)^{\mathrm{T}}$, which refers to the constant -1 in (1). The operator $\mathbf{Q}$ can be assembled as

$$\mathbf{Q} \;=\; p \cdot \mathbf{I} \;+\; q \cdot \mathbf{W}\mathbf{W}^{\mathrm{T}} \;+\; r \cdot \mathbf{W}^{\mathrm{T}}\mathbf{W} \;+\; s \cdot \mathbf{D}^{\mathrm{T}}\mathbf{D}\,,$$

where $\mathbf{I}$ is the identity matrix. The matrix $\mathbf{W}$ is given as

$$\mathbf{W} \;=\; \big(\mathbf{I} - \mathbf{G}\big)^{\lambda}\mathbf{H}\mathbf{G}\,,$$

such that $\mathbf{W}\mathbf{W}^{\mathrm{T}}$ and $\mathbf{W}^{\mathrm{T}}\mathbf{W}$ refer to (2) and (3), respectively. Then, $\mathbf{G}$ is a diagonal matrix with g_i, and $\mathbf{H}$ is sparse with $\mathbf{H}_{ij} = h_{ij}$ (see above). The matrix $\mathbf{D}$ is the backward difference operator that is applied to adjacent polyline segments in (4).

We use the reflective Newton method [Coleman and Li 1996] that is implemented in MATLAB's `quadprog`[1] function to solve the system. Due to frame coherence of the solutions, we obtain a significant speed-up by using the previous solution as an initial guess for the next solution. We perform the setup of the system matrix $\mathbf{Q}$ for the *next* time step in one CPU-thread, while another thread is dedicated to solving the system for the *current* time step, i.e., we pipeline the computation. Once a solution is available, it is streamed to the GPU to smoothly blend the current opacities to the latest solution. We use this fading in toward a new solution to prevent popping artifacts, as we do not get every frame a new solution.

5 Parameter Studies

5.1 Choosing g_i

As the meaning of *importance* depends on the application and goals of data exploration, we don't prescribe a particular importance function. Instead, we provide a generic way to incorporate *any* suitable importance measure, which is defined as g_i per polyline segment i. This enables the user to either explore the presence of, or to communicate a specific feature in the data. For all examples we use either the polyline length (i.e., all segments of a line have identical importance) or the segment curvature (i.e., integrated curvature). These measures are generic and available for any application. In addition, various methods exist that extract application-independent line features, for instance linear entropy [Furuya and Itoh 2008], angular entropy [Marchesin et al. 2010], scalar entropy fields [Xu et al. 2010], or the number of pixels that a line covers on the screen [Günther et al. 2011]. Alternatively, importance may be application-specific, e.g., distance to a tumor in medical imaging, cluster sizes in fiber tracking, or uncertainty in data. With our approach, we can incorporate any sophisticated importance function. Different choices for importance are shown for a synthetic tornado data set in Fig. 5 (line length) and Fig. 1 (curvature), which brings out the vortex core. (For both images we set $q = 1.2$, $\lambda = 5$.)

5.2 Optimization Parameters

The parameters p, q, r, s act as weights for the error terms, and λ emphasizes important lines. We set $p = 1$ and select q, r, s relatively to this value. The parameter q weights occlusion and is probably most important: it scales the amount of opacity in the final image. Setting $q = 0$ keeps all lines opaque (if r is – as described later – chosen relative to q.) An increasing q gradually fades out occluding lines. Effectively, this is the main tool to resolve occlusions introduced by lines in the foreground, as shown in Fig. 4(a): Here, we placed a camera behind a helicopter in slow forward flight close to the ground. The data is described in [Kutz et al. 2012]. The line

[1] We found that an equivalent formulation for MATLAB's `lsqlin` was slower, and same for quadratic/conic solvers in MOSEK (see `mosek.com`).

field represents air flow, and the direct visualization of the original data, shown in the top figure, suffers from vast occlusion. In the bottom figure, we use curvature as importance to reveal the main vortices: visible are not only the two vortices, released from the rotor blades and transported back, but also the vortex on the ground in front of the helicopter. The latter vortex is critical as it is the main source of small dust particles that cause hazardous brown-out conditions, which endanger ground personnel as well as passengers.

The parameter r has a more subtle effect: it accounts for fading out unimportant lines in the background that are occluded by important lines in the foreground. This effect is demonstrated in the flow around a short wall-mounted cylinder, simulated by [Frederich et al. 2008] and shown in Figure 4(b). Behind the interesting region there is an unimportant laminar layer that clutters the view for $r = 0$ (top). For $r > 0$ we obtain a much clearer visualization, as disturbing unimportant lines in the background are faded out (bottom). It is reasonable to select $r \leq q$, and our experiments show that setting r relative to q, as $r = \frac{1}{10}\, q$ is generally a good choice. We use this setting in the remainder of the paper unless stated otherwise.

The remaining parameter s weights a smoothing term that penalizes high variation of opacities along polylines. We always set $s = 0.3$.

Finally, the coefficient λ controls the emphasis of important lines. In Fig. 4(c), we demonstrate its utility in a hydrocyclone, a device to separate particles in a liquid suspension. If not stated otherwise, we set $\lambda = 1$ neutral, i.e., no increased emphasis on important lines.

6 Results and Discussion

We compare [Marchesin et al. 2010; Günther et al. 2011] and our algorithm in Figure 5. The columns show the initial line set (left) and the different methods. Data sets are arranged in the rows.

The first row shows magnetic field lines in the decay of magnetic knots, as present in astrophysical objects. Here, the topological reconnection of rings was studied by [Candelaresi and Brandenburg 2011], which temporarily form the shape of Borromean rings. The objective of the visualization is to bring out the highly occluded, symmetric rings, a task at which our technique performed best. We use line lengths as importance with $q = 2$, $r = 0.02$, and $\lambda = 3$.

The second row analyzes experimental wind tunnel data by [Yu et al. 2002] of a descending helicopter (viewed from below w/o the helicopter body). Of interest are the vortices that detach from the tips of the rotor blades: their impact on the rotor and on the noise generation (by blades cutting the vortices) should be analyzed. Our method produces clear views on the vortices. Here, curvature was used as importance, with $q = 2$, $r = 0.04$, and $\lambda = 2.5$.

The tornado data set shown in the third row is described in section 5.1. Since our approach is able to locally fade out segments, it can cover the entire viewport with lines (by using all lines) and at the same time reveal features like the tornado's vortex core (in contrast to [Marchesin et al. 2010]). Furthermore, this test case shows the advantage of being able to fade out line segments locally over using a global opacity value per line as in [Günther et al. 2011].

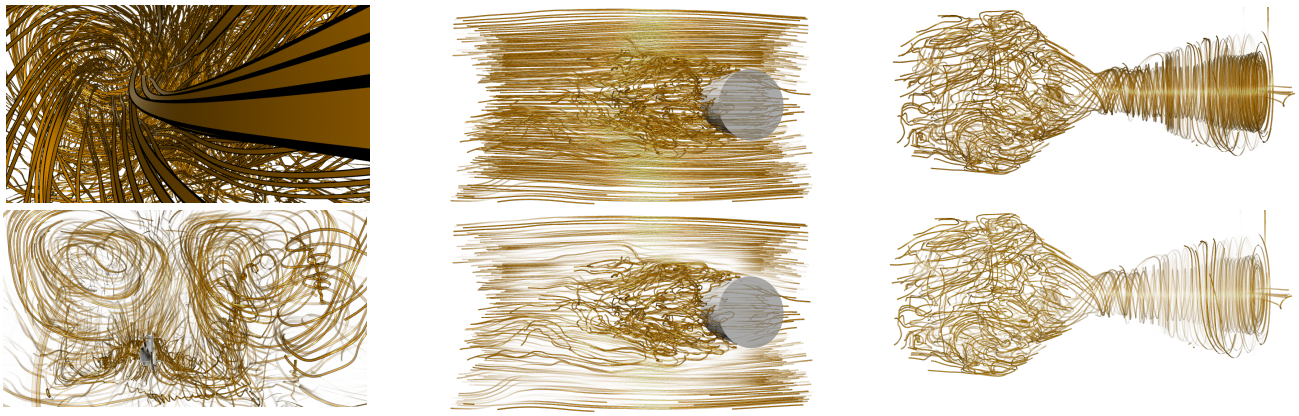
The fourth row shows medical data, i.e., blood flow through an aneurysm, simulated at the Institute of Fluid Dynamics and Thermodynamics, based on vessel geometry acquired at the University Hospital, both in Magdeburg. In the input data, the aneurysm is entirely occluded by flow through the blood vessel. An overview, taken from a distant viewpoint, is shown in Fig. 6(b). Not only the aneurysm but also convection due to wall reflection is made visible by our approach (curvature as importance and $p = 0.3$, $\lambda = 3.5$).

The fifth row shows a Rayleigh-Bénard convection, i.e., the forming of separate convection cells by fluids heated from below (simulated using the free software NaSt3DGP). The center left image in Fig. 1 depicts all four cells. To demonstrate the disadvantage of greedy algorithms, we placed the camera inside a convection cell. Both methods, [Marchesin et al. 2010] and [Günther et al. 2011], select highly occluding lines, while our approach reveals the occluded cells (curvature as importance and $q = 0.3$, $r = 0.06$, $\lambda = 3.5$).

Figure 6(a) shows visualizations of diffusion tensor data, i.e., tractography in medical imaging, which allows to infer the white-matter connectivity of the brain to diagnose vascular strokes or cancer. The DT-MRI data was acquired at the Central Hospital of Bremen. In Fig. 7, we visualize trajectories of three falling dice to depict motion. We show speed lines for every corner to demonstrate that our technique is also suited for automatic fading of lines in illustrative renderings. This scene is courtesy of Maik Schulze.

6.1 Performance

We measured the performance for an Intel Core i7-2600K CPU with 3.4 GHz, 16 GB RAM and a Nvidia GeForce GTX 560 Ti GPU with



(a) Air flow around helicopter. Top: input line set ($q = r = 0$). Bottom: Set $q = 0.2$, $r = 0.16$, $\lambda = 3$ to fade out unimportant parts.

(b) Flow in wake of a wall-mounted cylinder. Set $q = 0.4$, $\lambda = 1.5$. Top: visual clutter in background at $r = 0$. Bottom: resolved by $r = q/10$.

(c) Streamlines in a hydrocyclone. An increase of λ emphasizes important lines, i.e., the conical outflow part. ($q = 2$, top: $\lambda = 2.3$, bottom: $\lambda = 2$)

Figure 4: *Effect of parameters q, r and λ. Curvature is used as importance for all data sets.*

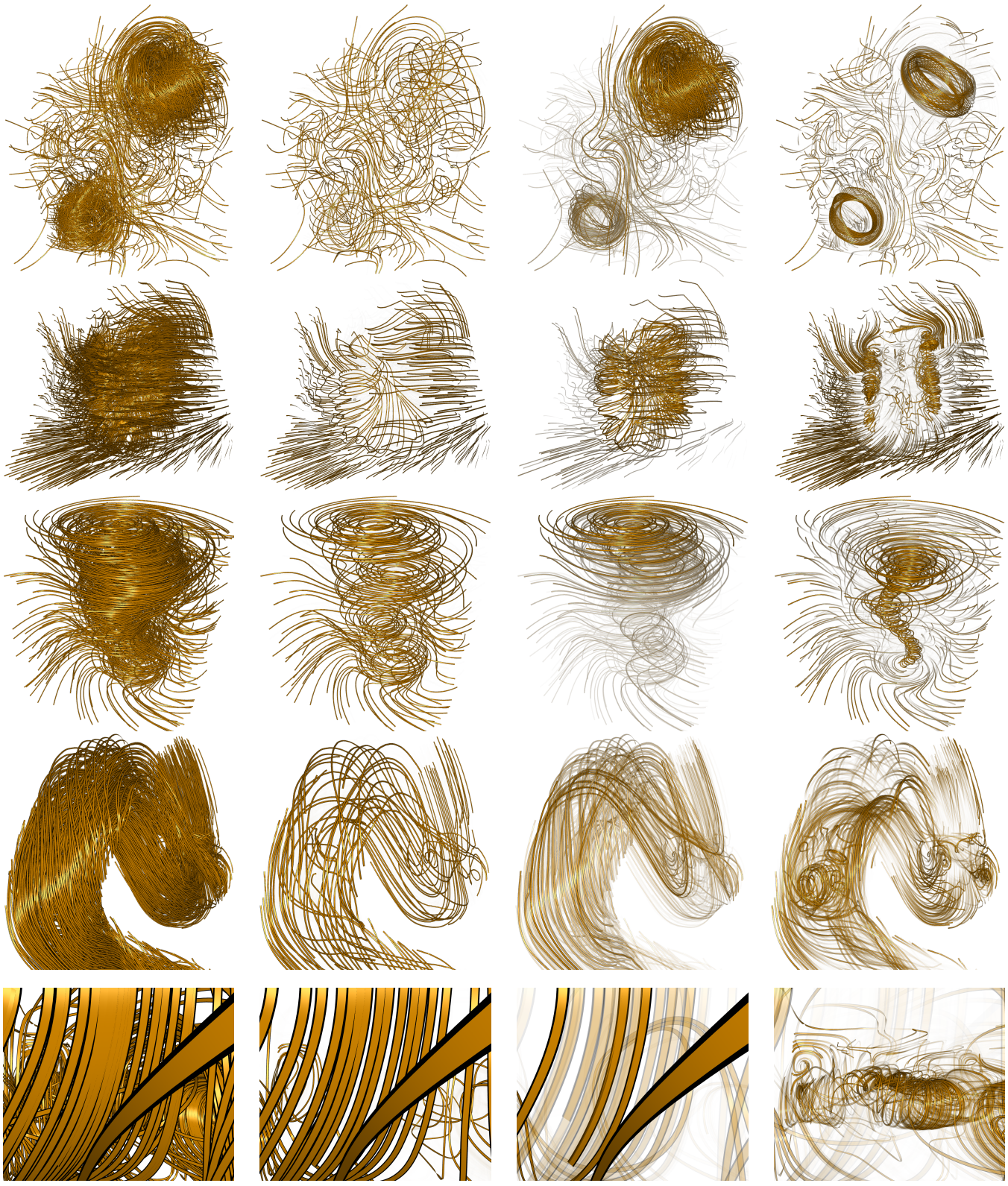
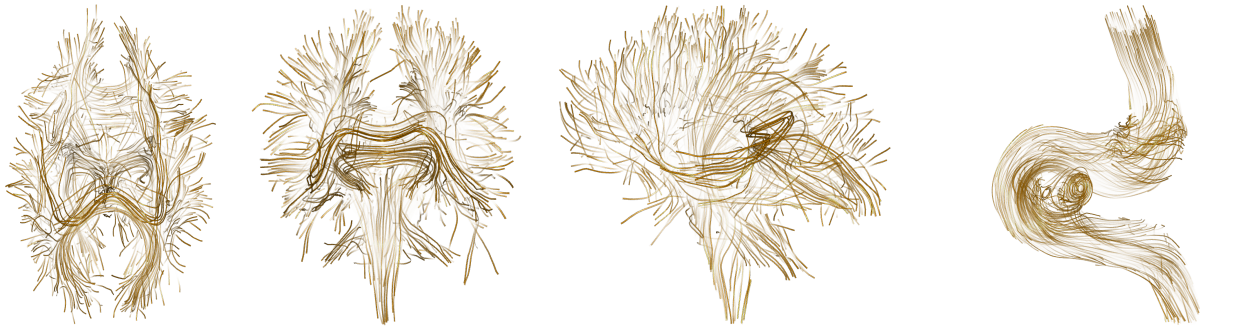


Figure 5: *Comparison to other line selection algorithms. Columns from left to right: input line set (i.e., all lines), [Marchesin et al. 2010], [Günther et al. 2011] and our approach. Data sets are in rows, top to bottom: Borromean rings, flow around a descending helicopter, tornado, aneurysm and Rayleigh-Bénard convection. Our method brings out specific features that are otherwise occluded, i.e., rings, vortex cores, the aneurysm (cf. Fig. 6(b)), and the convection cells (cf. Fig. 1).*



(a) line length, $p = 2$, $\lambda = 5$ (b) curvature, $p = 0.2$, $\lambda = 2.5$

Figure 6: *Tensor and flow data in medicial imaging: (a) Diffusion tensor lines. (b) Blood flow in vessel with an aneurysm.*

2 GB VRAM. The number of frames, h_{ij} assemblies, and solves per second are listed for all referenced figures in Table 2. Note that the assembling, solving, and rendering run in parallel. Thus, the slowest component poses the bottleneck, which was in all but two cases the solving. The only exceptions were the Rayleigh-Bénard convection in Fig. 5, due to the high number of transparent layers, and the dice in Fig. 7, for which the optimization problem was small so that the rendering was slower. The streaming of the fragment linked lists to the CPU took always 22 ms (due to the constant fragment pool size). All results were computed for a resolution of 1200×1000 with $16\times$ coverage-sampling anti-aliasing (CSAA).

Table 2: *Performance: Frames per second (Fps), assemblies of h_{ij} per second (Alg. 1) and solves per second (including computation of Q), with n being the number of segments. The bottleneck is **bold**.*

Data set	Figs.	Fps	Asm./s	Sol./s	n
Aneurysm	5	14.9	3.6	**1.0**	3,816
Aneurysm	6(b)	25.6	7.1	**1.4**	3,816
Bénard	1	13.1	5.6	**2.0**	2,224
Bénard	5	8.4	**1.4**	4.0	2,224
Borromean	1, 5	38.5	14.6	**3.3**	2,936
Brain (axial)	6(a)	57.1	15.2	**3.4**	5,808
Cylinder	4(b)	29.4	13.2	**11.4**	3,776
Dice	7	**128.2**	196.9	163.9	576
Heli Brownout	4(a)	26.0	7.9	**1.1**	8,136
Heli Descent	1, 5	19.3	10.8	**1.9**	6,896
Hydrocyclone	4(c)	22.8	7.6	**4.6**	2,016
Tornado	1, 5	27.7	11.4	**3.5**	2,648

6.2 Limitations

Our approach strives to keep a balance between information presentation and occlusion avoidance. There are two cases, shown in Figure 8, when it does not get meaningful results:

- There are *too many important structures*: for the example of streamlines in a turbulent flow, the whole domain may be densely covered with important lines, leading to a simple depth cueing as optimum.
- There are *too few important structures*: if all lines are of low interest (e.g., streamlines in a laminar flow), the optimum also gives a simple depth cueing.

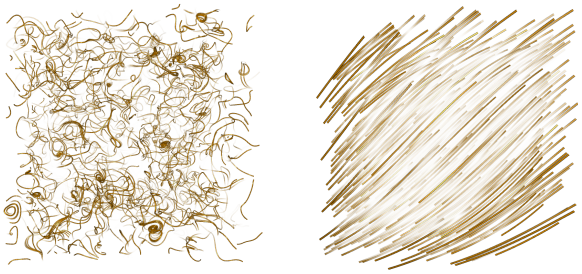
Although both cases can occur, we do not consider them as critical. For turbulent flows, as in the first case, streamlines are generally not an adequate approach for visual representation, since the shape of particular lines is dominated by randomness. In the second case, a high-quality rendering of a dense line field is not really necessary because of its simple structure.

Another limitation is the fact that our optimization does not explicitly incorporate the density of the lines. If many lines are spatially close to each other, the algorithm will assign similar opacities to them. In this case, a better result may be to assign one representative a high opacity and to make adjacent lines less visible. Since this leads to non-linear optimizations, we leave it to future research.

7 Conclusions

In this paper, we introduced the first method for line selection in 3D line fields that is based on a global optimization: We computed a globally optimal opacity for polyline segments by minimizing a quadratic error function, which formalizes desired properties (favor opaque lines, reduce occlusions, smooth opacity gradient along the lines). Our approach can be steered by a-priori domain knowledge, and it attains interactive, frame coherent and view-dependent visu-

Figure 7: *Trajectories of three falling dice. Each casts eight speed lines, which are partially faded out to reveal the helical trajectory. (curvature as importance, $p = 0.15$, $r = 0.1$, $\lambda = 2.5$)*



(a) Turbulent flow *with only* important lines. ($p = 4$, $r = 8$) (b) Laminar flow *without any* important lines. ($p = 0.1$, $r = 0.1$)

Figure 8: *Limitations: for extreme cases with no prominent feature to bring out, the results of our method resemble depth cueing.*

alizations. Moreover, it is the first approach that resolves occlusions effectively for every viewpoint position.

For the future, we plan to apply our line selection to unsteady line fields, for which the subdivision must then be modeled in a frame-coherent way. An automatic parameter setup can be further investigated for a faster exploration of large numbers of data sets. The setup of the system (Alg. 1) may benefit from using the GPU.

References

ANNEN, T., THEISEL, H., RÖSSL, C., ZIEGLER, G., AND SEIDEL, H.-P. 2008. Vector field contours. In *Proc. Graphics Interface*, 97–105.

CANDELARESI, S., AND BRANDENBURG, A. 2011. Decay of helical and nonhelical magnetic knots. *Phys. Rev. E 84*, 016406.

CHEN, Y., COHEN, J., AND KROLIK, J. 2007. Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics 13*, 1448–1455.

COLEMAN, T. F., AND LI, Y. 1996. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM J. on Optimization 6*, 4, 1040–1058.

EICHELBAUM, S., HLAWITSCHKA, M., AND SCHEUERMANN, G. 2013. LineAO – improved three-dimensional line rendering. *IEEE Transactions on Visualization and Computer Graphics 19*, 3, 433–445.

EVERTS, M. H., BEKKER, H., ROERDINK, J. B. T. M., AND ISENBERG, T. 2009. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics 15*, 1299–1306.

FREDERICH, O., WASSEN, E., AND THIELE, F. 2008. Prediction of the flow around a short wall-mounted cylinder using LES and DES. *Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM) 3*, 3-4, 231–247.

FURUYA, S., AND ITOH, T. 2008. A streamline selection technique for integrated scalar and vector visualization. In *IEEE Visualization Poster Session*.

GÜNTHER, T., BÜRGER, K., WESTERMANN, R., AND THEISEL, H. 2011. A view-dependent and inter-frame coherent visualization of integral lines using screen contribution. *Proc. Vision, Modeling, and Visualization (VMV)*, 215–222.

JOBARD, B., AND LEFER, W. 1997. Creating evenly-spaced streamlines of arbitrary density. *Proc. Eurographics Workshop on Visualization in Scientific Computing 7*, 45–55.

JOBARD, B., AND LEFER, W. 2001. Multiresolution flow visualization. *WSCG 2001 Conference Proceedings*, 33–37.

KUTZ, B. M., KOWARSCH, U., KESSLER, M., AND KRÄMER, E. 2012. Numerical investigation of helicopter rotors in ground effect. In *30th AIAA Applied Aerodynamics Conference*.

LEE, T.-Y., MISHCHENKO, O., SHEN, H.-W., AND CRAWFIS, R. 2011. View point evaluation and streamline filtering for flow visualization. In *Proc. IEEE Pacific Visualization*, 83–90.

LI, L., AND SHEN, H.-W. 2007. Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 630–640.

LI, L., HSIEN, H. H., AND SHEN, H. W. 2008. Illustrative streamline placement and visualization. *IEEE Pacific Visualization Symposium 2008*, 79–86.

LIU, Z., MOORHEAD, R., AND GRONER, J. 2006. An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics 12*, 965–972.

LUFT, T., COLDITZ, C., AND DEUSSEN, O. 2006. Image enhancement by unsharp masking the depth buffer. *ACM Trans. Graph. 25*, 3, 1206–1213.

MA, J., WANG, C., AND SHENE, C.-K. 2013. Coherent view-dependent streamline selection for importance-driven flow visualization. *Proc. SPIE 8654, Visualization and Data Analysis*.

MARCHESIN, S., CHEN, C.-K., HO, C., AND MA, K.-L. 2010. View-dependent streamlines for 3D vector fields. *IEEE Transactions on Visualization and Computer Graphics 16*, 1578–1586.

MATTAUSCH, O., THEUSSL, T., HAUSER, H., AND GRÖLLER, E. 2003. Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In *Proc. Spring Conference on Computer Graphics (SSCG)*, ACM, 213–222.

MAULE, M., COMBA, J. L., TORCHELSEN, R. P., AND BASTOS, R. 2011. A survey of raster-based transparency techniques. *Computers & Graphics 35*, 6, 1023–1034.

MCLOUGHLIN, T., JONES, M., LARAMEE, R., MALKI, R., MASTERS, I., AND HANSEN, C. 2012. Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics*.

MEBARKI, A., ALLIEZ, P., AND DEVILLERS, O. 2005. Farthest point seeding for efficient placement of streamlines. In *IEEE Visualization*, 479–486.

TAO, J., MA, J., WANG, C., AND SHENE, C. 2013. A unified approach to streamline selection and viewpoint selection for 3D flow visualization. *IEEE Transactions on Visualization and Computer Graphics 19*, 393–406.

TURK, G., AND BANKS, D. 1996. Image-guided streamline placement. In *Proc. SIGGRAPH*, 453–460.

VERMA, V., KAO, D., AND PANG, A. 2000. A flow-guided streamline seeding strategy. In *IEEE Visualization*, 163–170.

WANG, C., AND SHEN, H.-W. 2011. Information theory in scientific visualization. *Entropy 13*, 1, 254–273.

XU, L., LEE, T.-Y., AND SHEN, H.-W. 2010. An information-theoretic framework for flow visualization. In *IEEE Transactions on Visualization and Computer Graphics*, 1216–1224.

YANG, J. C., HENSLEY, J., GRÜN, H., AND THIBIEROZ, N. 2010. Real-time concurrent linked list construction on the GPU. *Computer Graphics Forum 29*, 4, 1297–1304.

YE, X., KAO, D., AND PANG, A. 2005. Strategy for seeding 3D streamlines. *IEEE Visualization Conference*, 471–478.

YU, Y., TUNG, C., VAN DER WALL, B., PAUSDER, H.-J., BURLEY, C., BROOKS, T., BEAUMIER, P., MERCKER, Y. D. E., AND PENGEL, K. 2002. The HART-II test: Rotor wakes and aeroacoustics with higher-harmonic pitch control (HHC) inputs – the joint German/French/Dutch/US project. *American Helicopter Society 58th Annual Forum*.

YU, H., WANG, C., SHENE, C.-K., AND CHEN, J. 2012. Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8, 1353–1367.

ZÖCKLER, M., STALLING, D., AND HEGE, H.-C. 1996. Interactive visualization of 3D vector fields using illuminated stream lines. In *IEEE Visualization*, 107–113.