EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

Hierarchical Opacity Optimization for Sets of 3D Line Fields

Tobias Günther, Christian Rössl and Holger Theisel

Visual Computing Group, University of Magdeburg

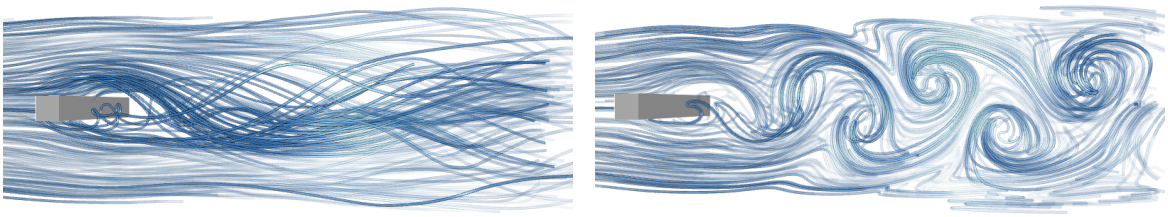

Figure 1: *View-dependent, frame-coherent and interactive selection of meaningful lines for sets of line fields: pathlines (left, 58 ms/opacity update) and streaklines (right, 78 ms/opacity update) in a cylinder flow.*

Abstract

The selection of meaningful lines for 3D line data visualization has been intensively researched in recent years. Most approaches focus on single line fields where one line passes through each domain point. This paper presents a selection approach for sets of line fields which is based on a global optimization of the opacity of candidate lines. For this, existing approaches for single line fields are modified such that significantly larger amounts of line representatives are handled. Furthermore, time coherence is addressed for animations, making this the first approach that solves the line selection problem for 3D time-dependent flow. We apply our technique to visualize dense sets of pathlines, sets of magnetic field lines, and animated sets of pathlines, streaklines and masslines.

This is the authors preprint. The definitive version is available at http://diglib.eg.org/ and http://onlinelibrary.wiley.com/.

1 Introduction

Line fields are families of curves that cover a 3D domain densely. This means that for every point in a 3D domain there is exactly one line passing through it. For their visualization, the selection of lines to be rendered has been identified to be the most crucial problem. Several line selection approaches have been proposed in recent years. In many applications, not one single line field but sets of line fields occur, meaning that more than one line passes through a point in 3D. These sets of line fields depend on one or multiple parameters. Examples are pathlines, streaklines and masslines of time-dependent flow, as well as sequences of instantaneous magnetic field lines.

We present the first global approach to line selection in sets of line fields. We follow the insight by Günther et al. [GRT13] that line selection should be a view-dependent global optimization process. This is because the representation of information in 3D has to be carefully traded for occlusion avoidance. However, Günther et al. only presented a solution for single line fields. Its direct extension to sets of line fields is not straightforward for two reasons:

- Sets of line fields consist of significantly more lines than a single field. Since view-dependent line selection is an interactive process, the approach in [GRT13] has to be accelerated to cope with larger amounts of line segments.
- Parametric sets of line fields can be naturally visualized by an animation, i.e., a time sequence of line fields. In this case, time coherence has to be ensured.

This paper presents solutions to both problems. We solve the first problem using a hierarchical representation of the lines to make the problem size of the numerical optimization view-dependent. The second problem is addressed by an additional temporal smoothness term in the underlying energy minimization together with a propagation of opacity values to lines in adjacent time steps. We apply our technique to a number of parametric line fields, including pathlines, streaklines and masslines in time-dependent flow fields, as

well as sequences of magnetic field lines. In particular, this is the first paper in flow visualization presenting a global line selection algorithm for 3D pathlines, streaklines and masslines. Fig. 1 shows an example: the display of pathlines and streaklines in a time-dependent cylinder flow.

2 Related Work

The rendering of integral lines, such as streamlines, pathlines or streaklines, is a fundamental and well-established tool in geometric flow visualization [MLP*10] and an active area of research [BCP*12]. A particular problem is the placement and selection of the lines to be displayed. This section reviews algorithmic solutions to this problem for steady vector fields and unsteady flow.

2.1 Line Selection in Steady Vector Fields

Much work went into the placement of streamlines in 2D domains or 2D manifolds [TB96, JL97, VKP00, JL01, MAD05, LMG06, LHS08, SLCZ09]. The goal of these approaches is to find suitable seeds for streamlines such that, e.g., an as even as possible spacing is achieved. There are direct extensions to the 3D case that search for optimal seeds based on streamline density [MTHG03], their features [YKP05, YWSC12], or similarity measures [CCK07, MJL*12]. Having in mind the rendering of a 2D image from 3D data, the line placement problem becomes significantly more complex in three dimensions: there are intersections and occlusions of projected lines, effects that are *view-dependent*.

To cope with this, Li and Shen [LS07] introduced an image-space approach that projects lines seeded in 2D back into the scene. The generated lines do not cross in image space, but there is no guarantee that they cover and represent the entire field. To overcome this, Li and Shen proposed to merge line sets from different views, but they do not address the arising occlusion problem. Marchesin et al. [MCHM10] presented the first approach that considered occulusion in the scene by using a greedy selection of lines from a precomputed set of streamlines. Their selection algorithm prunes the initial line set iteratively based on local importance measures and the overlap that lines produce after projection to the screen. The greedy nature of this approach prohibits frame coherence, i.e., lines appear or disappear instantly when changing the view. Recent interactive approaches to visualize 3D vector fields follow the line *selection* paradigm rather than the seed structure optimization.

To ensure that a selection is representative, Xu et al. [XLS10] consider the conveyed information compared to the entropy of the input vector field. Their approach does not address occlusion, but it provides an informed choice for the initial line set in approaches that typically start with random lines. Ma et al. [MWS13a] addressed frame coherence by combining a static set of lines that projects the main information in all views, and a view-dependent line set that is selected to provide continuity between adjacent viewpoints. This combination reduces popping artifacts between frames, but they are not completely avoided. A different approach to view-independent line rendering by McLoughlin et al. [MJL*12] uses new, fast similarity measures for an interactive line clustering.

Yu et al. [YWSC12] propose hierarchical streamline bundling, a view-independent approach to streamline clustering. Chen et al. [CYY*11] first partition the streamlines with high entropy, then form clusters, and finally render representative lines with illustrative techniques. A recent interactive exploration framework by Ma et al. [MWS13b] uses a graph representation of the line field to assist in clustering and detecting spatial and hierarchical relationships.

Also related are approaches for automatic camera placement and camera path generation, which are particularly useful for the automated analysis of large sequences of data. Lee et al. [LMSC11] apply a maximum intensity projection of the entropy field and use this view-dependent metric for the evaluation and prioritization of streamlines that should project significant entropy while minimizing occlusion at the same time. This greedy approach is not frame-coherent, but it finds the optimal static viewpoint with respect to their metric. Tao et al. [TMWS13] combine the selection of best lines and best viewpoints in an information-theoretic approach to generate optimal camera paths. Here, the loci of viewpoints and camera paths are restricted to a sphere.

When aiming at unrestricted, explorative navigation in 3D line fields, the line selection algorithm must be view-dependent, interactive, occlusion-aware, and frame-coherent. The recent work by Günther et al. [GRT13] suggests that the selection should also be importance-driven in a sense that it always provides a clear view onto the scene parts that are relevant to the user. All this holds equally for the case of unsteady flow fields.

2.2 Line Selection in Unsteady Flow

There is considerably less work on line (placement and) selection for unsteady flow data in 2D and 3D domains likewise. For 2D, Jobard and Lefer [JL00] suggested to depict a coherent animation of streamlines instead of displaying pathlines. For each streamline a corresponding line in the next time step is found by integrating candidate lines from sample points in the next time step and choosing the closest line. Weinkauf et al. [WTS12] proposed a selection strategy for creating static illustrations of pathlines and streaklines. Visual clutter is reduced by avoiding cusps in the projections of the lines. Until now, the selection problem was neither addressed for animated pathlines nor for streaklines.

For 3D line fields, Günther et al. [GBWT11] proposed a view-dependent and frame-coherent approach, which sets the opacity of a line based on its contribution to the viewport, i.e., the number of pixels visible in the rendered image.

Since this is a local approach, the method is potentially fast enough for line selection from sets of line fields. However, in a follow-up work, Günther et al. [GRT13] showed that even for steady data this screen contribution blending method is of limited use, because occlusion is not considered in the selection process. So far, there exists no satisfying solution that allows explorative navigation in sets of line fields, such as pathlines or streaklines.

3 Background and Contribution

This work is based on and extends the approach by Günther et al. [GRT13], which formulates line selection in the steady case as a global optimization problem with the goal of showing as much information as possible while avoiding occlusion. We briefly review this opacity optimization approach in this section and provide an overview of the novel contribution required for the unsteady case, i.e., visualizing sets of line fields such as pathlines or streaklines.

3.1 Opacity Optimization

Günther et al. [GRT13] divide polylines into n segments for which opacity values α_i (with $i \in 1,\dots,n$) are computed as minimizers of a quadratic energy E that penalizes occlusion. This is based on importance information g_i given per polyline segment and occlusion degrees h_{ij} that specify how much one segment i occludes another segment j.

A central role plays the energy E, consisting of four terms:

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

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

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

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

with *bounded* variables $0 \leq \alpha_i \leq 1$ (0 for transparent and 1 for opaque). The first term (1) inherently makes all polyline segments as visible as possible by penalizing the variation of α_i from 1. Thus, if no occlusions occur, all segments are opaque. The second term (2) fades out segments i that are unimportant ($1-g_i$ is high) and occlude to a certain degree $h_{ij} > 0$ segments j that are important (g_j is high). Reducing α_i means fading out an unimportant segment in front of an important segment. The third term (3) is symmetric to the second term in the sense that it removes unimportant polyline segments *behind* important ones in order to remove background clutter and to produce perception-enhancing halos around the important structures. In both terms, the parameter λ controls the emphasis of the important structures by steering the fall-off of g_i from 1. The last term (4) is a smoothness term that penalizes opacity variation along a polyline by considering adjacency of segments a_{ij}. The weights p,q,r,s of the individual terms balance their respective contribution. A guideline for their selection was discussed in [GRT13], which we followed.

Two measures are essential for expressing the energy above: The occlusion degree h_{ij} is determined automatically by rasterization. The importance g_i of the pieces is a user-defined weight function that allows the domain expert to focus on structures that are currently relevant to the visual exploration. These can be simple application-independent metrics like length or curvature of lines, or more advanced measures like linear entropy [FI08] or angular entropy [MCHM10]. For all example images we used curvature as importance.

3.2 Problem Statement and Novel Contribution

The minimization of energy E requires the solution of a bounded-variable least-squares problem. The numerical solution dominates the computation time. It is decoupled from interaction and rendering such that approximately one asynchronously computed solution per second is sufficient for opacity updates in the interactive rendering. For the interactive exploration of line sets, Günther et al. [GRT13] report a maximum problem size $n < 10,000$ polyline segments; this bound may vary depending on the data (and the desired rate of solves), $n \approx 3,000$ is a typical value. This is not sufficient for the interactive exploration of parametric sets of line fields such as pathlines or streaklines, because these fields consist of many more, possibly time-dependent, lines.

As a first contribution, we extend the opacity optimization such that the problem size n is variable *depending on the current view* and decoupled from the number of input line segments, which we denote as c. The goal is to ensure a feasible problem size n even for large values of c. We achieve this by a hierarchical representation of lines that is used for an adaptive opacity sampling (Sec. 4).

Second, we show how to achieve temporal coherence for sets of line fields in the presence of time-dependent data. This requires an extension of the energy term as well as an interpolation of opacities across hierarchy levels (cf. Sec. 5).

4 Hierarchical Opacity Optimization

The bottleneck in the original opacity optimization [GRT13] is typically the numerical energy minimization. The rate at which solutions are computed depends on the number of unknowns, i.e., the total number n of polyline segments used for opacity optimization. We aim at increasing this rate drastically by making n a view-dependent function of a potentially large total number of candidate segments c. In all our examples, we use $c = 20k$.

Generally, the line representation used for rendering is decoupled from the representation used for solving the numerical problem. In [GRT13] this is achieved by resampling each

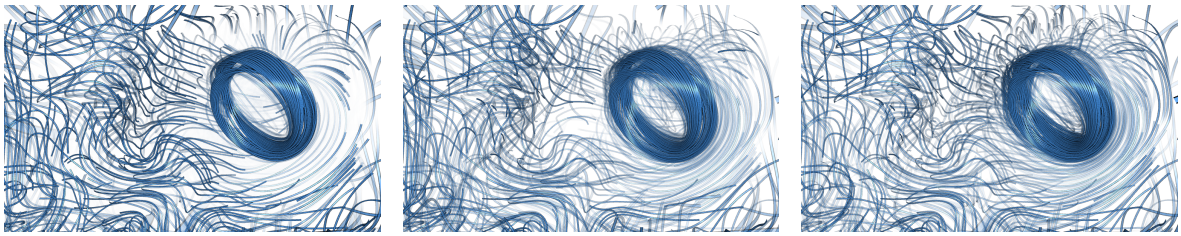

Figure 2: BORROMEAN RINGS. *A better distribution and higher number of polyline segments yield details around important structures, here, the field lines close to the ring. Left: original opacity optimization [GRT13] for $n = 6k$ with $920\,ms/solve$. Center: novel hierarchical approach for $n = 1.4k$ with $239\,ms/solve$, and right for $n = 4k$ with $629\,ms/solve$.*

polyline by only few linear segments of equal length. This coarse representation is used in the numerical optimization, and the computed opacity values are then interpolated on the original polyline. The resampling solely depends on the arc length. Even though the resampled line typically consists of only few—at most 8—polyline segments the total number of segments n can become high, as all lines are treated equally regardless of their contribution to the image.

The novel idea is to advance from a static arc length-based resampling to a mapping that yields coarsely sampled polylines by an adaptation to the view. This way, not only the problem size is decreased, but effectively a better distribution of opacity samples is achieved in a sense that lines closer to the camera obtain a locally higher line sampling rate, see Fig. 2. We achieve this by a hierarchical representation of polylines that enables an efficient extraction of coarse, adaptively sampled polylines used for the opacity optimization.

4.1 Hierarchical Polyline Representation

In an off-line preprocess, we build a hierarchical representation for each polyline in the input set of lines as follows. The initially given polyline is assumed to be sampled uniformly with respect to the arc length, possibly at a high sampling rate. These uniform segments represent the finest level of the hierarchy. The coarser levels are built iteratively in a bottom-up approach: for the next coarser level, every pair of adjacent segments in the current level is collapsed into one single segment. For an uneven number, the remaining segment is left as is. Doing this recursively until there remains only one root segment yields a binary tree with leaf nodes referring to the original segments and internal nodes referring to recursively collapsed segments. Finally, doing this for all lines, the union of all line representations defines a *forest* of binary trees. If the input is a set of line fields, we process every field in the set individually and obtain a set of forests (used in Sec. 5). Fig. 3 illustrates the setting of a single line field containing two lines. Details are given in Sec. 6.2.

4.2 View-dependent Adaptation of Problem Size

From the hierarchical representation of lines as a forest of binary trees, we determine the discretization of polylines for the numerical optimization dynamically based on the current view. The selection of an appropriate set of resampled lines refers to finding a cut through the forest. An ideal cut should make the projected lengths of the joined polyline segments, i.e., the nodes' footprints, as equal as possible. The rationale of the heuristic is to obtain an as uniform as possible discretization of opacity samples. Note that while a low cut yields fine granularity of opacity samples, a higher cut would still maintain a fair distribution of samples, which improves visual quality (see Fig. 2). We use a breadth-first search to efficiently find a good approximation to such a cut. Thereby, each node is associated with the bounding box of the represented linear segment. Starting from the root nodes of the trees, we traverse toward the leaf nodes with two stopping criteria: First, the traversal stops at a node when the bounding box of this node is completely outside the view frustum. In this case, the node is "trivially" rejected and neither considered for the opacity optimization nor the rendering. Second, the traversal stops when a node reached a desired viewport footprint. The footprint threshold f is a user parameter that balances the solve rate (and hence interactive response) versus the number of possible opacity changes along the lines. The footprint of a leaf node is approximated by assuming straight lines per node. Then, for small boxes the diagonal of the screen space bounding box of the projected bounding box of the node is a practical approximation. For internal nodes the footprint of the children is summed up. Fig. 3 illustrates the finding of a cut conceptually. Fig. 4 visualizes a cut for $f = 30\%$ viewport diagonal.

The view-dependent selection is particularly helpful in large scenes, for instance the blood vessel in Fig. 5. There, only about 33% of the polyline segments are visible. As shown in the figure, the view-dependent selection of polyline segments from a large set of candidates produces a better

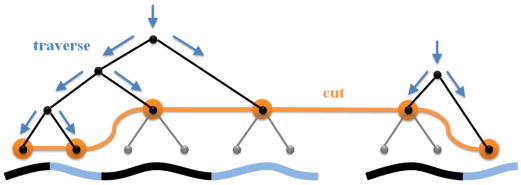

Figure 3: *Illustration of the hierarchical representation (Sec. 4.1) for two lines. For finding a cut (orange nodes) the tree is traversed (blue arrows) top-down (Sec. 4.2).*

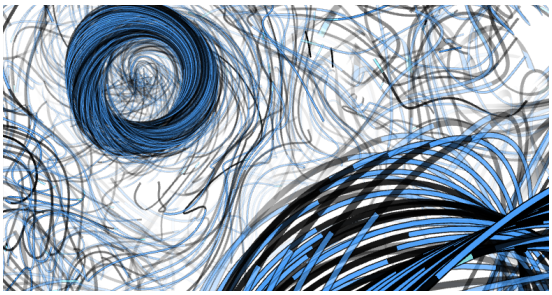

Figure 4: *Visualization of a cut with the cut segments color-coded (alternating blue/black). Note that more polyline segments are spent in the foreground. ($n = 2.5k$ segments).*

set for the optimization compared to the fixed, precomputed subdivision. For using computed opacity values as an initial guess in the next solving iteration, the opacities are propagated in the hierarchy from the old cut to the new cut. Fig. 6 illustrates this and the aforementioned culling of segments.

5 Processing Parametric Sets of Line Fields

Our goal is to visualize sets of line fields that may depend on parameters such as time and/or mass. Examples are animated pathlines and streaklines, as shown in Figure 1, as well as masslines. For an overview of the *space of pathlines*, there is an alternative to animation: Rendering *all* lines in a static scene and letting the opacity optimization put focus on the most important events of the entire sequence. In this case, time is an additional dimension which leads to a larger amount of data. This case can be handled effectively by the hierarchical optimization as is.

5.1 Enforcing Temporal Coherence in Opacity Values

Typically, the visualization of parametric sets of line fields is an animation of lines over time. As an example, Figure 7 shows a sequence of time steps of a streakline animation. This requires to solve an additional problem: the sequence of animation frames should be coherent over time. Thereby, we have to carefully distinguish between coherent opacity values on the lines—this is the problem we solve—and the possibly incoherent behavior of the lines themselves.

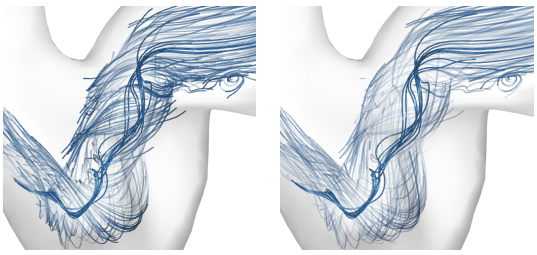

Figure 5: BYPASS. *Vortex core in a blood vessel. Left: standard opacity optimization ($n = 8k$ at $2039\,ms/solve$). Right: novel approach provides clearer view at the vortex core (cut contains $n = 1.8k$ polyline segments at $191\,ms/solve$).*

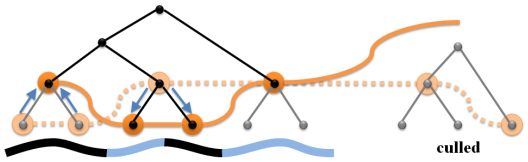

Figure 6: *Propagation of opacity values between cuts in the hierarchy. Thereby, invisible polyline segments are culled. Dashed line is the old cut, solid line is the new cut.*

The latter means that lines may suddenly disappear, when line parts leave the domain or when integral lines suddenly change their direction due to an insufficient temporal resolution of the animation. This behavior is inherent to the data and should therefore not be faded out (even though it may be due to artifacts, e.g., from sampling).

In order to ensure temporal coherence of opacity values, we modify the global energy: instead of E, we minimize

$$E_t = E + u \sum_{i=1}^{n} \left(\alpha_i - \alpha_i'\right)^2 .$$

The new term penalizes deviation from the solution α' of the previous iteration and hence introduces "inertia" to suppress sudden changes. Note that thereby inertia depends on the solving rate. The new parameter u balances smoothness of the animation versus accuracy in a sense of deviation from the minimizer of original energy E. The choice $u = 1$ (w.r.t. the normalization $p = 1$ suggested in [GRT13]) showed good results in all our experiments.

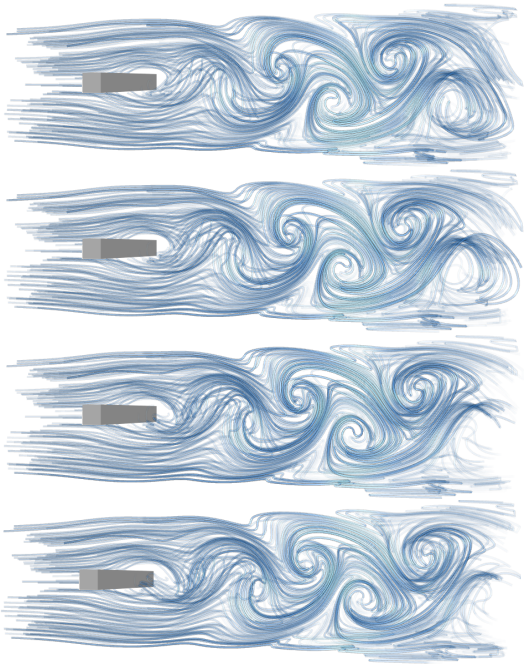

Figure 7: *Time steps of the evolution of streaklines forming a vortex street behind a* SQUARE CYLINDER. *Time advances from top to bottom.*

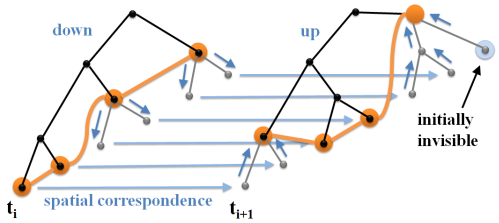

Figure 8: *For propagating opacity to the next time step a correspondence at the leaf nodes is required. Polyline segments without a corresponding segment in the previous time step are initialized as invisible, i.e.,* $\alpha_i = 0$.

5.2 Inter-frame Interpolation of Opacity Values

To improve the performance of the numerical solver, we provide the previously computed opacity values α' as initial guess to the next solving iteration. As mentioned in Section 4, we represent time-parametric sets of line fields as sequence of hierarchical representations, i.e., one forest per time step. Consequently, we have to interpolate opacity values not only within cuts in one forest but between two forests: Then, a correspondence problem arises for the transport of opacities α_i from nodes of the current cut to a possibly topologically different cut in the next time step. Our approach is visualized in Fig. 8. First, we propagate opacity values in the current time step down to the leaf nodes. From there, opacities are copied to leaf nodes in the next time step, i.e., the next forest. Since leaf nodes represent polyline segments of roughly equal length, leaf nodes of the same index can be assumed to be spatially close to each other. If a line has grown and no corresponding segment exists, its opacity value is set to 0, i.e., "invisible". Then, opacity values are propagated upward from the leaf nodes to the new cut.

5.3 Hiding Latency of the Solver

In the original opacity optimization [GRT13], opacity values of the polyline segments fade slowly toward the newly computed solution to hide the latency of the solver. Fading opacity in animated sequences requires their projection onto the line set in the next time step, i.e., a correspondence has to be established. Depending on the type of integral curve, a meaningful correspondence is naturally given, i.e., opacity can be copied to the subsequently released particle (e.g., pathlines or streaklines) or it can be advected with the vertices (e.g., timelines or masslines). Thus, we store the current opacity values used for display not per node (polyline segment), but as vertex attribute. We acquire the opacity values to fade to by interpolation of solutions from nearby polyline segments. This final step is similar to the original algorithm.

6 Implementation and Technical Details

6.1 Initial Line Set

For the display of all pathlines, we place seeds on the space-time boundaries to ensure a dense sampling. Animated pathlines and streaklines were seeded in inlet regions. Masslines in the HELICOPTER data set are released from the sediment bed, as they depict sand particle behavior. Clustering lines and choosing representatives [CYY*11, YWSC12, MJL*12] is a possible improvement to reduce the size of the input set required to represent the field. Given the line set, we divide all lines into a total of c polyline segments such that all segments have roughly equal lengths. We use $c = 20k$ in all our examples. Afterwards, the forests are formed bottom-up for each line field individually.

6.2 Matrix Setup Using the Hierarchy

We made a few detailed modifications due to the hierarchical data structure when setting up the system matrix. Counting the occlusion degrees $h_{i,j}$ requires to identify the corresponding polyline segment (i or j) for each fragment. Günther et al. [GRT13] proposed a parameterization that in our case yields the index of the polyline segment at the base level. In addition, for each base level polyline segment, we maintain a pointer to the selected cut node. For identifying whether cut nodes are adjacent ($a_{i,j}$), the left-most and right-most children of the cut nodes are tested for neighborhood. The indices of these children are precomputed for each node. For propagating opacities (and initially the importance g) in the hierarchy, the entries of child nodes are averaged.

6.3 Decoupling Rendering and System Setup

Günther et al. [GRT13] used a single rendering pass of *all lines* to acquire fragment linked lists that were used for both rendering and occlusion degree extraction on the CPU. We found that splitting them into two passes led to a significant frame rate improvement (see later in Sec. 7.2). In the first pass, we construct the linked lists at quarter viewport resolution and with single sampling. Using these shorter, separate lists to compute approximations of the occlusion degrees reduces the amount of data to copy to the CPU and the time spent on iterating them when counting occlusion pairs. Moreover, the subsequent high quality rendering pass does no longer require to render all lines, but only polyline segments in the view frustum with an opacity $\alpha_i > 0$.

7 Results

In this section, we show examples for visualizations generated by the novel hierarchical approach, and we discuss its performance. As we are processing time-dependent data, we would also like to point to the accompanying **video**.

7.1 Visualization of Steady and Time-dependent Data

SQUARE CYLINDER. We applied the novel hierarchical opacity optimization to a number of time-dependent data sets. Figure 1 shows the unsteady wake of a square cylinder by animated pathlines and streaklines—the latter revealing a vortex street. The uniformly resampled version of

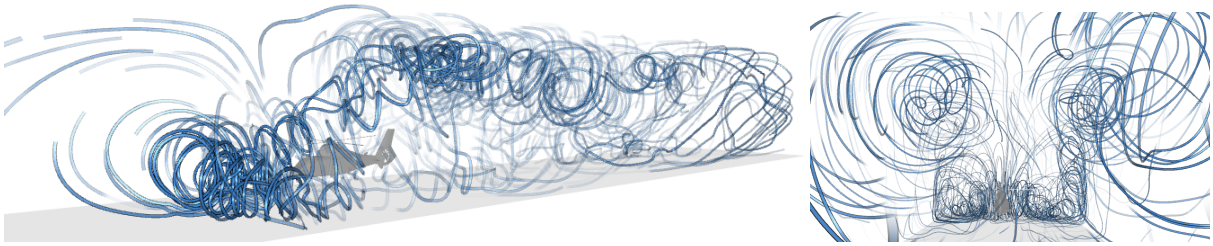

Figure 9: *Animated pathlines in the* HELICOPTER HOVER *sequence from outside and inside the domain. The location of the vortex on the ground is best seen in the left image (*$83\,ms/solve$, $n = 1k$*). In the image on the right, the camera looks at the two vortices in the air. There, the vortex on the ground is visible in the distance (*$135\,ms/solve$, $n = 1k$*).*

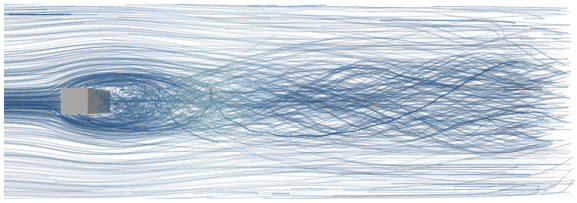

Figure 10: *Set of all pathlines in the* SQUARE CYLINDER *sequence. The opacity optimization shows an overview of events for any point in time (*$437\,ms/solve$, $n = 2k$*).*

this vector field sequence was provided by Tino Weinkauf (see [vFWTS08]) and is based on the Navier-Stokes simulation by Camarri et al. [CSBI05]. The set of all pathlines is shown in Fig. 10, depicting events from any point in time.

HELICOPTER HOVER. The second unsteady sequence captures the simulation of the flow regime around a helicopter in slow forward flight close to the ground. The data by Kutz et al. [KKKK12] was studied for brown-out conditions [GKKT13]. Fig. 9 depicts the data from two perspectives. For this data, masslines were used to analyze relationships of inertial sand particles. A massline connects particles of different mass reached after a certain time, cf. [GKKT13]. In Fig. 11 a vortex ring is visible that stirs up particles from the loose sediment bed, hindering the view of the pilot.

WALL-MOUNTED CYLINDER. Frederich et al. [FWT08] simulated the air flow around a short wall-mounted cylinder. The view on animated pathlines in Fig. 12 in the wake of the cylinder is hindered by a laminar layer that is removed by the opacity optimization.

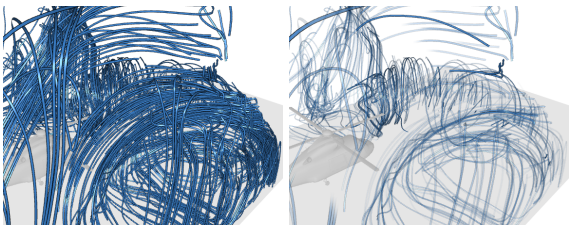

Figure 11: *Masslines forming a vortex ring in the* HELICOPTER *sequence. Left: cluttered input line field, right: our method (*$78\,ms/solve$, $n = 612$*).*

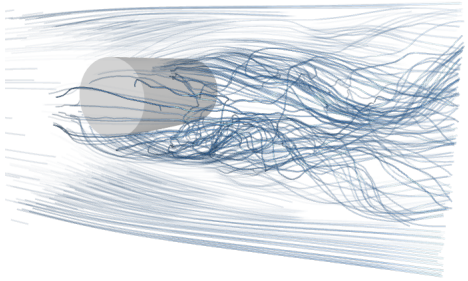

Figure 12: *Wake of a* WALL-MOUNTED CYLINDER *shown using animated pathlines (*$139\,ms/solve$, $n = 1k$*).*

BORROMEAN RINGS. If the spatial relation of events permits it, individual time steps of an unsteady sequence can be shown simultaneously in one static view, which consequently contains an increased number of lines. Candelaresi and Brandenburg [CB11] simulated the decay of a magnetic field, with field lines initially interlocked in the shape of Borromean rings. During the simulation the rings decay to two small magnetic knots that drift apart, visualized in Fig. 13.

HELICOPTER DESCENT. The following steady data sets were used to demonstrate the improved performance (Sec. 7.2) and visual quality of the novel hierarchical optimization. In Fig. 14 (left), streamlines depict the air flow around a descending helicopter (viewed from below without the helicopter geometry). The velocity field stems from experimental wind tunnel data by Yu et al. [YTvdW*02].

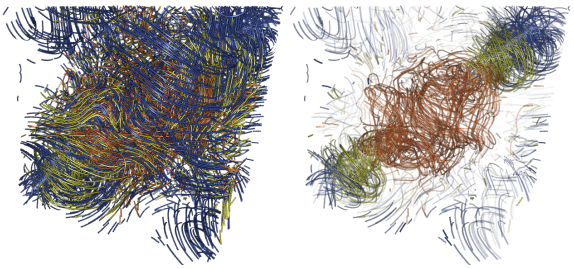

Figure 13: BORROMEAN RINGS. *Multiple magnetic fields from a time series of a magnetic flux decay simulation. The time steps are color-coded from orange to yellow to blue. Left: input line fields, right: visualization of the two knots therein moving apart (*$244\,ms/solve$, $n = 3.5k$*).*

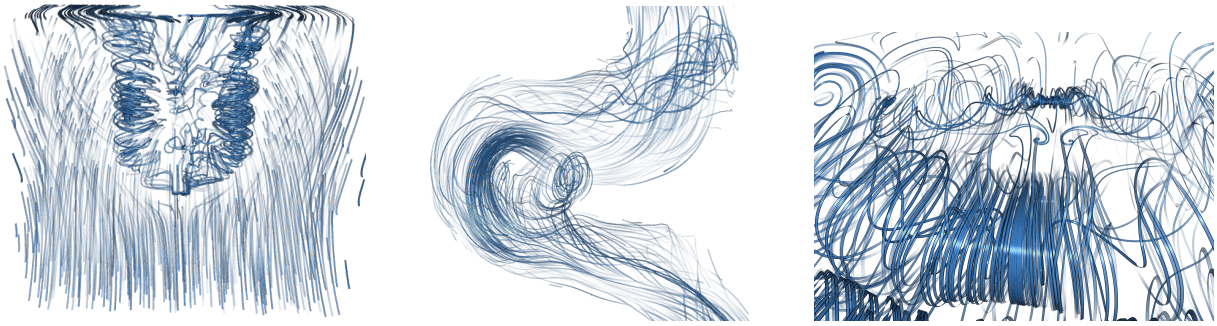

Figure 14: *Results in steady data sets. From left to right: Vortices behind a* HELICOPTER *in descent (*$99\,ms/solve$*,* $n = 2k$*), blood flow through an* ANEURYSM *of a pathological vessel (*$129\,ms/solve$*,* $n = 2k$*), and field lines in a* RAYLEIGH-BÉNARD CONVECTION *(*$228\,ms/solve$*,* $n = 1.4k$*).*

Blood flow in BYPASS and ANEURYSM. As an example for medical applications, we use hemodynamics data acquired at the Leipzig Heart Center and the University Hospital of Magdeburg (processed at Institute of Fluid Dynamics and Thermodynamics in Magdeburg). We visualize blood flow in an aortic bypass (Fig. 5), measured by PCMRI and in an aneurysm at a pathological vessel (Fig. 14, center).

RAYLEIGH-BÉNARD CONVECTION. In Fig. 14 (right), the camera is placed inside a Rayleigh-Bénard convection that arises if a thin layer of liquid is heated from below. The opacity optimization clears the view on the cells. The velocity field was simulated using the free software NaSt3DGP.

7.2 Performance

For the following performance measurements, we used an Intel Core i7-2600K CPU with 3.4 GHz and an Nvidia GeForce GTX 560 Ti GPU. All results were computed for a resolution of 1200×1000 with $16\times$ coverage-sampling anti-aliasing (CSAA).

The hierarchy construction took 4–8 ms per line set, which corresponds to 50–60 % of the total preprocessing time. Timings for our sequences and an average time per set (which corresponds to the steady case) are listed in Table 1. The hierarchy data structure contains parent and child pointers, bounding boxes, importance values and several pointers for faster access, see 6.2. For 500 lines, each consisting of 40 segments, this yields less than 2.5 MB in total per line set.

We compared the solve rate and frame rate of the original method [GRT13] with the novel hierarchical approach in Figs. 15 and 16 for the benchmark data shown in Figs. 5 and 14. We observed a $1.4–6.8\times$ speedup for rendering and a $4.3–9.7\times$ speedup for solving while maintaining a similar visual quality for the generated images. Using the standard method, the solver provided at least one solution per second for almost all data sets, which matches the results reported by Günther et al. The only exception was the blood flow in a BYPASS data set in Fig. 5, in which nearby vessel geometry is filled with irrelevant streamlines. Generally, as soon as the camera moves into the domain (e.g., into the vessel's bounding box), the speedup factor for solving increases due to the culling in the view-dependent selection of polyline segments. (Note that *culling* in this context refers to decreasing the problem size for the solver.)

Table 1: *Preprocessing timings for the* SQUARE CYLINDER *(streak lines),* WALL-MOUNTED CYLINDER *(path lines) and* HELICOPTER HOVER *(mass lines) sequences.*

Preprocessing	Sq. Cyl.	Wall-mt.	Helicopt.
Number of sets	200	400	250
Average per set	14.04 ms	6.97 ms	10.60 ms
Total time	2.81 s	2.79 s	2.65 s

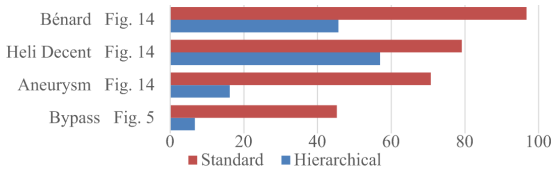

Figure 15: *Rendering time in milliseconds.*

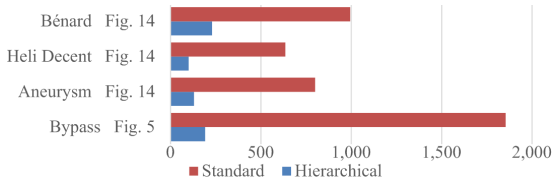

Figure 16: *Solving time in milliseconds.*

Next, we evaluate the use of the previous solution as initial guess for the next solve. For this, we quantify the speedup for solving and the cost for propagation of opacities between solves for a *still camera*. Then, in the steady case, the given initial solution is a "perfect" guess, which thus results in the best possible performance. However, this performance is generally unreachable for animated line fields, where the system of equation changes. But it states an upper bound that we strive to approach. Not using an initial guess resulted in a worse performance. The timings for solving in Table 2 indicate that the propagation has a significant effect and yields a constant speedup of 10–30 ms in our tests.

Table 3 summarizes timings for the different steps of the algorithm for both rendering and solving (including setup of the system matrix), which are executed in different threads. Times are averages for one complete turn-over in the threads' main loops. In the following, the steps and their bottlenecks are discussed. The fading toward new opacity solutions is fast and primarily depends on the number of vertices in the scene. The speed of the low resolution pass depends on the image overdraw. Incidentally, the time for assembling the system matrix goes up, as more time is required to compute the occlusion degrees. The high quality rendering pass, however, does not need to render all the lines. Thus, it has less overdraw and is therefore faster. It depends on the number of segments chosen by the solver, which in turn depends on the user parameters of the energy. The computation time for the cut depends on the number of segments to begin with and the recursion depth. The propagation of opacity values from the previous frame depends on the number of segments. The same holds for the solving performance. If the input line set is larger, while targeting the same problem size, fewer line segments are availble per line. In summary, the time-critical parts are the system setup and the solving. The first scales with the number of input lines and the latter depends on the number of polyline segments chosen in the cut.

The total solving performance is suitable for a visualization of animated line fields. For most scenes, we were below 100 ms and observed up to 139 ms only in scenes with high overdraw. For steady scenes, we pushed the number of lines and polyline segments, thus, the solving time went up. Our worst case example was the BORROMEAN RINGS scene in Fig. 2 with up to 629 ms. There, we have shown that a higher cut at 239 ms produced better results than the original method [GRT13] with 920 ms. Compared to [GRT13], we perform significantly better with a speedup factor near 10.

Table 2: *Benefits of an initial guess (on/off) for the next solve iteration. Shown are the solver turn-over times in milliseconds for the* SQUARE CYLINDER, WALL-MOUNTED CYLINDER *and* HELICOPTER HOVER *data, each for a fixed view.*

Test Case	Sq. Cyl.	Wall-mt.	Helicopt.
Steady	54.93	85.84	64.14
Unsteady (on)	55.15	99.52	76.10
Unsteady (off)	62.27	129.14	97.59

Table 3: *Performance of the different algorithm steps for the* SQUARE CYLINDER, WALL-MOUNTED CYLINDER *and* HELICOPTER HOVER *data (all unsteady) with timings given in milliseconds. The upper part is executed in the rendering thread, the lower part is executed in the solver thread.*

Pipeline Step	Sq. Cyl.	Wall-mt.	Helicopt.
Segments n (cut)	2.5k	1.4k	1k
Alpha Fading	0.01	0.01	0.01
Low Res Pass	0.66	2.53	1.94
HQ Pass	12.22	10.03	10.81
Turn-Over	**12.89**	**12.57**	**12.76**
Compute Cut	0.89	0.34	0.72
Assemble Matrix	35.48	52.94	33.75
Propagate α'	0.06	0.08	0.08
Solve	18.72	46.16	41.56
Turn-Over	**55.15**	**99.52**	**76.10**

7.3 Discussion and Limitations

We demonstrated that the proposed hierarchical opacity optimization enables interactive exploration of sets of line fields. For meeting the higher performance demands of animated line field exploration, the novel method does not sacrifice visual quality for speedup—quite the contrary. The improved polyline discretization for the opacity computation leads to an improved rendering quality at a smaller problem size. However, this makes it somewhat difficult to faithfully compare the standard and hierarchical methods: for the same problem size n they yield generally different results. In our experiments, the effective values of n are usually lower for the hierarchical approach in the comparisons. The magnitude of n is steered by the maximum footprint threshold (Sec. 4.2), which is the user parameter that defines the visual quality achieved by the optimization. We do not see much benefit in having a true "load balancing" in a sense of finding the best cut for a prescribed n: first, even a greedy search for such cut is more expensive. Second, a dynamic adaptation is preferable in practice, such that the footprint threshold is higher during interaction (low n) and lower at a user-defined value when the interaction pauses (high n).

Our method shares the limitations of the standard opacity optimization: adequate input data is required such that all relevant features are represented. A local oversampling could be addressed in a preprocess, e.g., using a clustering approach on lines and visualizing cluster representatives. Furthermore, the animation of parametric line sets requires a correspondence between lines. This is not always given, for instance, in time-dependent magnetic fields.

8 Conclusions

We extended the work by Günther et al. [GRT13] so that sets of 3D line fields can be visualized at interactive rates. The main contributions consist in the therefore required design of a hierarchical representation of line sets and a modification of the energy term together with an efficient interpolation of opacity values. The first enables a view-dependent and feasible problem size for opacity optimization even though the amount of input data may be magnitudes larger than for "steady" line sets. In addition, the opacity sampling is regularized, which improves the visual quality for a similar problem size. The second provides temporal coherence of opacity in rendering and for guessing initial solutions. With this, we proposed the first global line selection approach for pathlines, streaklines and masslines in 3D flow.

References

[BCP*12] BRAMBILLA A., CARNECKY R., PEIKERT R., VIOLA I., HAUSER H.: Illustrative flow visualization: State of the art, trends and challenges. *Eurographics State-of-the-Art Reports* (2012), 75–94. 2

[CB11] CANDELARESI S., BRANDENBURG A.: Decay of helical and nonhelical magnetic knots. *Phys. Rev. E 84*, 1 (2011), 016406. 7

[CCK07] CHEN Y., COHEN J., KROLIK J.: Similarity-guided streamline placement with error evaluation. *IEEE TVCG 13* (2007), 1448–1455. 2

[CSBI05] CAMARRI S., SALVETTI M.-V., BUFFONI M., IOLLO A.: Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers. In *XVII Congresso di Meccanica Teorica ed Applicata* (2005). 7

[CYY*11] CHEN C.-K., YAN S., YU H., MAX N., MA K.-L.: An illustrative visualization framework for 3D vector fields. *Computer Graphics Forum 30*, 7 (2011), 1941–1951. 2, 6

[FI08] FURUYA S., ITOH T.: A streamline selection technique for integrated scalar and vector visualization. In *IEEE Visualization Poster Session* (2008). 3

[FWT08] FREDERICH O., WASSEN E., THIELE F.: 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 (2008), 231–247. 7

[GBWT11] GÜNTHER T., BÜRGER K., WESTERMANN R., THEISEL H.: A view-dependent and inter-frame coherent visualization of integral lines using screen contribution. *Proc. Vision, Modeling, and Visualization (VMV)* (2011), 215–222. 2

[GKKT13] GÜNTHER T., KUHN A., KUTZ B., THEISEL H.: Mass-dependent integral curves in unsteady vector fields. *Computer Graphics Forum (Proc. EuroVis) 32*, 3 (2013), 211–220. 7

[GRT13] GÜNTHER T., RÖSSL C., THEISEL H.: Opacity optimization for 3D line fields. *ACM Transaction on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 120:1–120:8. 1, 2, 3, 4, 5, 6, 8, 9

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. *Proc. Eurographics Workshop on Visualization in Scientific Computing 7* (1997), 45–55. 2

[JL00] JOBARD B., LEFER W.: Unsteady flow visualization by animating evenly-spaced streamlines. *Computer Graphics Forum (Proc. Eurographics) 19*, 3 (2000), 31–39. 2

[JL01] JOBARD B., LEFER W.: Multiresolution flow visualization. *Proc. WSCG* (2001), 33–37. 2

[KKKK12] KUTZ B. M., KOWARSCH U., KESSLER M., KRÄMER E.: Numerical investigation of helicopter rotors in ground effect. In *30th AIAA Applied Aerodynamics Conference* (2012). 7

[LHS08] LI L., HSIEN H. H., SHEN H. W.: Illustrative streamline placement and visualization. *Proc. IEEE PacificVis* (2008), 79–86. 2

[LMG06] LIU Z., MOORHEAD R., GRONER J.: An advanced evenly-spaced streamline placement algorithm. *IEEE TVCG 12*, 5 (2006), 965–97. 2

[LMSC11] LEE T.-Y., MISHCHENKO O., SHEN H.-W., CRAWFIS R.: View point evaluation and streamline filtering for flow visualization. In *Proc. IEEE PacificVis* (2011), pp. 83–90. 2

[LS07] LI L., SHEN H.-W.: Image-based streamline generation and rendering. *IEEE TVCG 13*, 3 (2007), 630–640. 2

[MAD05] MEBARKI A., ALLIEZ P., DEVILLERS O.: Farthest point seeding for efficient placement of streamlines. In *IEEE Visualization* (2005), pp. 479–486. 2

[MCHM10] MARCHESIN S., CHEN C.-K., HO C., MA K.-L.: View-dependent streamlines for 3D vector fields. *IEEE TVCG 16*, 6 (2010), 1578–1586. 2, 3

[MJL*12] MCLOUGHLIN T., JONES M. W., LARAMEE R. S., MALKI R., MASTERS I., HANSEN C. D.: Similarity measures for enhancing interactive streamline seeding. *IEEE TVCG 19*, 8 (August 2012), 1342–1353. 2, 6

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum 29*, 6 (September 2010), 1807–1829. 2

[MTHG03] MATTAUSCH O., THEUSSL T., HAUSER H., GRÖLLER E.: Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In *Proc. SSCG* (2003), pp. 213–222. 2

[MWS13a] MA J., WANG C., SHENE C.-K.: Coherent view-dependent streamline selection for importance-driven flow visualization. *Proc. SPIE 8654, Visualization and Data Analysis* (2013). 2

[MWS13b] MA J., WANG C., SHENE C.-K.: Flowgraph: A compound hierarchical graph for flow field exploration. In *Proc. IEEE PacificVis* (2013), pp. 233–240. 2

[SLCZ09] SPENCER B., LARAMEE R. S., CHEN G., ZHANG E.: Evenly spaced streamlines for surfaces: An image-based approach. *Computer Graphics Forum 28*, 6 (2009), 1618–1631. 2

[TB96] TURK G., BANKS D.: Image-guided streamline placement. In *Proc. SIGGRAPH* (1996), pp. 453–460. 2

[TMWS13] TAO J., MA J., WANG C., SHENE C.: A unified approach to streamline selection and viewpoint selection for 3D flow visualization. *IEEE TVCG 19* (2013), 393–406. 2

[vFWTS08] VON FUNCK W., WEINKAUF T., THEISEL H., SEIDEL H.-P.: Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE TVCG 14*, 6 (November 2008), 1396–1403. 7

[VKP00] VERMA V., KAO D., PANG A.: A flow-guided streamline seeding strategy. In *IEEE Visualization* (2000), pp. 163–170. 2

[WTS12] WEINKAUF T., THEISEL H., SORKINE O.: Cusps of characteristic curves and intersection-aware visualization of path and streak lines. In *Topological Methods in Data Analysis and Visualization II*, Mathematics and Visualization. Springer Berlin Heidelberg, 2012, pp. 161–175. 2

[XLS10] XU L., LEE T.-Y., SHEN H.-W.: An information-theoretic framework for flow visualization. *IEEE TVCG 16*, 6 (2010), 1216–1224. 2

[YKP05] YE X., KAO D., PANG A.: Strategy for seeding 3D streamlines. *IEEE Visualization* (2005), 471–478. 2

[YTvdW*02] YU Y., TUNG C., VAN DER WALL B., PAUSDER H.-J., BURLEY C., BROOKS T., BEAUMIER P., MERCKER Y. D. E., PENGEL K.: 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* (2002). 7

[YWSC12] YU H., WANG C., SHENE C.-K., CHEN J.: Hierarchical streamline bundles. *IEEE TVCG 18*, 8 (2012), 1353–1367. 2, 6