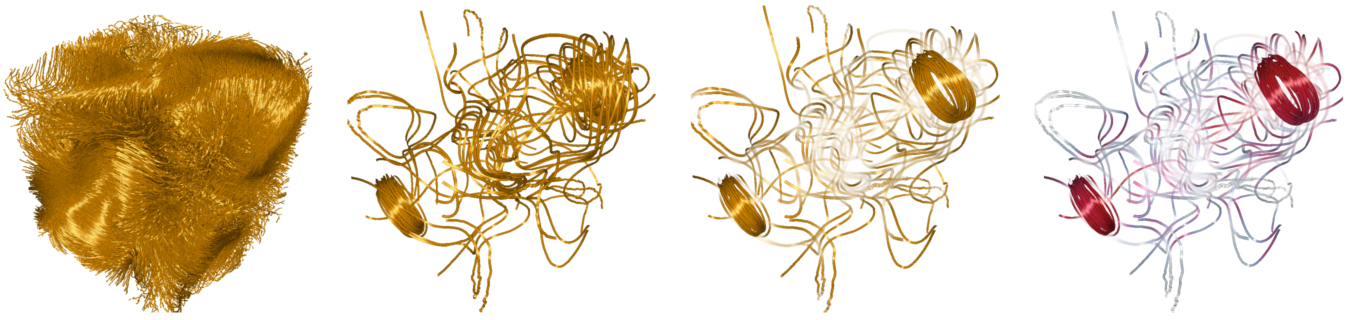
Vision, Modeling, and Visualization (2024)
L. Linsen and J. Thies (Eds.)

Optimization of Opacity and Color for Dense Line Sets

Berkan Tuncay and Tobias Günther

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany



(a) *Line Extraction* **(b)** *Line Clustering* **(c)** *Opacity Optimization* **(d)** *Color Optimization*

Figure 1: *This figure gives an overview of our interactive visualization pipeline. (a): First, a dense set of line data is generated, which is equipped with importance values $g(t)$ and coloration scalar values $c(t)$, which are later color mapped. (b): To reduce the line set, agglomerative hierarchical clustering is applied to the line dataset, comprising distance metric calculation and representative selection. (c): Using decoupled opacity optimization [GTG17], the opacity is optimized to maximize the visibility of important structures. (d): Taking the varying opacity along the lines into account, the color map is adjusted via histogram equalization to utilize the full range of colors.*

Abstract

In flow visualization, the depiction of line geometry in three-dimensional domains is often accompanied by occlusions. If there is a notion of which geometry is important to see, then a careful adjustment of the transparency is possible to ensure that irrelevant geometry is not occluding the meaningful structures, which is an inherently view-dependent problem. Past work in this line of research focused on the view-dependent adjustment of the transparency only and left the color channel open for the encoding of additional information. For a given viewpoint, the colormap could be set by the user, but once the view changes, the visible geometry is different and the colormap might no longer be utilizing its full color range. Thus, in this paper, we readjust the color transfer function to the new view, such that the colors of the colormap are utilized uniformly in the final image. To this end, a visibility histogram of all scalar values is recalculated and equalized on the GPU each frame. Further, past approaches required a set of lines that is not too dense, since the opacity optimization would otherwise fade out all lines similarly. For this reason, we incorporate a hierarchical line clustering for which we experimentally study the influence of distance metrics, linkage options, and representative choices. We apply the method in a number of scientific data sets, including examples from atmospheric sciences, aerodynamics, and electromagnetism.

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ ***Scientific visualization;*** • ***Computing methodologies*** $\rightarrow$ ***Visibility;***

1. Introduction

In flow visualization, streamlines are commonly used to display the behavior of particles in a vector field [MLP*10, SBGC20]. When the domain is three-dimensional, lines often occlude each other making it easy to overlook structures of interest. Given a notion of importance for each point along the lines, opacity optimization [GRT13, GTG17] adjusts the transparency of the lines to clear the view onto relevant structures. In this paper, we extend the approach in two ways. First, when the line density is too high, all lines cause some amount of occlusion and thus opacity optimization tends to fade out all lines similarly, not keeping a subset of lines as representatives. Thus, we apply clustering in a preprocess, which is commonly used to adjust the line density [KFW16, KW18]. We utilize agglomerative hierarchical clustering, for which we experiment with distance metrics, linkage methods, and representative choices. Second, opacity optimization intentionally only adjusted the transparency of the lines and kept the color channel open for the encoding of additional information. We encode an additional scalar property using a color map. However, which part of the line geometry is visible depends on the view, and hence the resulting utilization of the color map range is also view-dependent. Inspired by visibility histograms [CM10] and multi-scale adjustment of color maps [WWLM*19], we first form a histogram of the visible scalar value ranges and then perform a histogram equalization to arrive at a color map that maximizes the utilization of the available color map range. Our view-dependent color adjustment enhances relative differences, but inhibits the reading of absolute values. To retain semantic structures, a part of the color map can be fixed to maintain the meaning of certain value ranges. In summary, we add two extensions to the decoupled opacity optimization algorithm [GTG17]:

1. We use agglomerative hierarchical clustering to reduce the set of lines, such that occlusions can be resolved meaningfully.
2. We form a visibility histogram of the scalar values and adjust the color map to maximize the used color map range.

The algorithm is applied in several line data sets from atmospheric sciences, aerodynamics, and electromagnetism.

2. Related Work

In the following, we cover related work on streamline placement and clustering, the optimization of their visibility, and their color.

Line Placement and Clustering. Streamlines are curves that follow the flow tangentially. In 2D flows, they are commonly placed with uniform density. An early density-based optimization of streamlets was proposed by Turk and Banks [TB96]. Jobard and Lefer [JL97] proposed a greedy algorithm that seeds streamlines a separation distance away from existing lines and traces them until they get closer than a testing distance. Later, this approach was accelerated by Liu et al. [LMG06]. To guarantee a good representation of flow features, Verma et al. [VKP00] started with a template-based seeding around critical points. A farthest-point sampling of seed points was proposed by Mebarki et al. [MAD05]. In addition to uniform placements, a local adjustment of the streamline density is possible [SHH*07]. For 3D flows, evenly-spaced placements have been generated by Mattausch et al. [MTHG03]. If the density is too high, occlusion becomes a problem. To reduce the number of lines, clustering is a common choice [Jai10], which requires similarity metrics between trajectories. Zhang et al. [ZHT06] compared a number of line distance metrics, including (point-wise) Euclidean distance, PCA+Euclidean, Hausdorff, dynamic time warping, and longest common subsequences. Yu et al. [YWSC11] clustered streamlines based on saliency and their geometric distance. Rössl and Theisel [RT12] segmented the domain using streamline similarity. In the context of hemodynamics, Oeltze et al. [OLK*14] compared distance metrics and proposed the reduced mean closest point distance between streamlines, which we use later. Kanzler et al. [KFW16] reduced the line density by a minimum cost perfect matching to obtain a balanced line hierarchy. Kanzler and Westermann [KW18] proposed an interactive brush for the exploration of cluster hierarchies. Han et al. [HTW20] learnt a similarity between streamlines and surfaces in latent space, which was used for exploration. For a comprehensive introduction to streamline seeding and placement methods, we refer to the survey of Sane et al. [SBGC20].

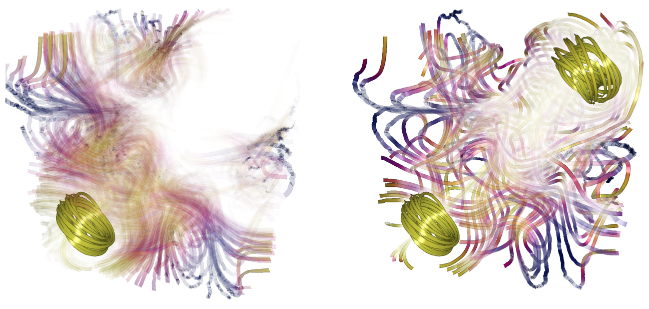

(a) *no clustering* **(b)** *with clustering*

Figure 2: *Opacity optimization fails when too many lines are optimized, since all lines cause occlusion. By using agglomerative hierarchical clustering, the line set size is reduced. Here, for the* BORROMEAN *with parameter* $\lambda = 1.1, q = 90$ *and* $r = 120$.

Visibility Optimization. Visibility optimization strives for an improved visibility of important structures in the domain, for which there are two orthogonal approaches: either the transparency (or thickness) of lines is adjusted or the viewpoint is improved. Early view-dependent approaches considered the pixel occupancy of the drawn streamlines, adding them one-by-one in a greedy manner [MCHM10]. A popular choice for measuring the information content are entropy-based measures, that were considered in viewpoint and streamline selection [LMSC11]. The coherence between viewpoints was studied to improve coherence during navigation [MWS13]. Instead of seeding lines, Günther et al. [GRT13] adjusted the transparency of streamlines for a given camera position by phrasing a linear optimization problem that considered how discrete pieces of lines were occluding each other. The approach has later been extended to time-dependent line geometry [GRT14] and to surface geometry [GSME*14]. For volume data, Ament et al. [AZD17] have shown that the linear optimization has a closed-form solution in ray space when the object-space smoothing is done in a post-process. This has afterwards been translated back to point, line, and surface geometries [GTG17], leading to the decoupled opacity optimization that this paper is building up on. More recently, the unknowns of the optimization have been stored in Fourier domain [BRGG20] and with moments [ZRPD20]. For volume data, the transmittance optimization has been combined with viewpoint optimization [HG24]. In opacity optimization [GRT13], the optimization tends to fade out bundles of equally-important lines equally rather than deciding on one representative to keep, see Fig. 2. Thus, one aspect of this paper is to adjust the line density adaptively via an agglomerative hierarchical clustering to obtain a sufficiently sparse set of lines. Such a clustering has previously been used to adjust the line discretization [GRT14], but it was not yet used in an opacity optimization to cluster the lines.

Coloring and Stylization. A number of approaches are available to encode information along streamlines. First of all, information can be mapped to color [FTC20, WNW*22]. The spatial perception of lines can be improved by adding shading [ZSH96, MPSS05]. Stoll et al. [SGS05] rendered tubes instead, on which the flow direction can be indicated by textures. The efficient ray tracing of tube primitives on the GPU was discussed by Han et al. [HWU*19]. To better convey the depth order of lines, halos have been added to lines [EBRI09]. The local twisting of the field can be shown by using colored ribbons called streamtapes [CYY*11]. Neuhauser et al. [NWKW22] concentrated on the efficient rendering of ribbons. Global illumination effects such as ambient occlusion [EHS13] in dense and transparent tube renderings [KRW18, GG21] improve the depth perception. Kern et al. [KNM*20] compiled a benchmark of rendering techniques for large numbers of transparent lines. In this paper, we adjust the color map such that the full range is used in the view-dependent visualization. In direct volume rendering, Correa and Ma [CM10] computed and visualized visibility histograms, which convey how much a scalar value appears in the image. Waldin et al. [WWLM*19] adjusted the color mapping in multi-scale data to the current view. Both approaches inspired our adjustment of the color map to the view-dependent changes of the visible geometry.

3. Color and Visibility Optimization of Dense Line Sets

In this paper, we aim to visualize a vector field in conjunction with a scalar field, as shown in Fig. 1. For this, streamlines are employed to depict the flow field and the scalar field information is encoded through the use of color. To begin, it is necessary to extract line data from the flow field. Streamlines are generated by numerically solving the initial value problem for a set of random seed points. Subsequently, scalar field values are sampled at each line vertex and are added to the line data. These scalar values are used to determine the opacity and coloring of a given line segment and are henceforth referred to as importance and coloration scalar, respectively. Next, an agglomerative hierarchical clustering technique is applied to the set of lines. It involves computing pairwise distances between individual lines, performing clustering through a bottom-up merging process, and then selecting representatives for each resulting cluster. The opacity optimization method is employed to assign transparency values to each line, aiming to maximize the visibility of important lines. It entails solving an energy minimization problem, which encourages high opacity for all lines while penalizing occlusion or cluttering of important lines by accordingly adjusting the transparency values. Finally, color values are assigned to all lines in a manner that maximizes the utilization of the colormap. This is accomplished by performing a histogram equalization on the histogram of visible coloration values. Thereby, a subset of the color map can be fixed by the user, such that value ranges of semantic structures can be preserved, e.g., the zero reference in a diverging color map, or a specific structure receiving an exclusive color.

3.1. Line Extraction

First, we extract streamlines to represent the vector field. Along the lines, two scalar fields are user-defined, which are later used for deciding the opacity and for assigning a line color.

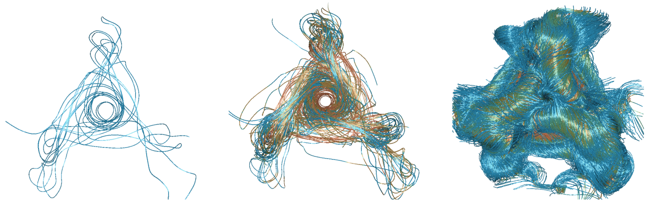

(a) *3 clusters* **(b)** *30 clusters* **(c)** *2000 clusters*

Figure 3: *Different number of clusters in agglomerative hierarchical clustering in the* TREFOIL *data set. With too few or too many clusters, structures are missed or redundant, respectively.*

Streamline Tracing. Given is a vector field $\mathbf{v}(\mathbf{x}) : \mathcal{D} \to \mathbb{R}^3$ defined over the domain D. Streamlines $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^3$ are curves that follow the flow tangentially:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{v}(\mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{1}$$

where $\mathbf{x}_0$ is the seed point. We seed the streamlines uniformly in the domain to achieve a high line density. A suitable subset is later selected by clustering. The lines are integrated forward and backward using a fourth-order Runge-Kutta integrator and trajectories are terminated once they reach a source or sink, exit the domain, or reach a maximum length.

Scalar Selection. The user provides two scalar fields. One scalar field denotes the line importance $g(t) : \mathbb{R} \to [0,1]$, the other is a scalar that is later mapped to color $c(t) : \mathbb{R} \to [0,1]$. Both functions are chosen according to the needs of the application. Following Günther et al. [GRT13], we provide a number of defaults that can be computed from the vector field, such as the arc length of the line, the line curvature, or the vorticity magnitude. In the absence of the vector field, the measures above could be computed from trajectories directly [FTG24]. Both functions, $g(t)$ and $c(t)$, are normalized to map to $[0,1]$. For the importance, $g(t) = 0$ means low importance and $g(t) = 1$ means high importance.

3.2. Line Clustering

Line clustering requires a notion of distance between lines to decide which ones to group together. To account for the two scalars $g(t)$ and $c(t)$ in the similarity metric, we append them with a user-weighted scale (λ_g, λ_c) to the curve $\mathbf{x}(t) = (x(t), y(t), z(t))^{\mathrm{T}}$:

$$\bar{\mathbf{x}}(t) = \big(x(t), y(t), z(t), \lambda_g g(t), \lambda_c c(t)\big)^{\mathrm{T}} \tag{2}$$

Given two discretized lines $\bar{\mathbf{x}}_i$, $\bar{\mathbf{x}}_j$, a number of distance metrics $d(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$ are available. We utilize the reduced mean closest point distance (rMCPD) [OLK*14], which is:

$$d(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \min\big(d'(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j), d'(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_i)\big) \tag{3}$$

$$d'(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \mathrm{mean}_{\mathbf{s}_i \in \bar{\mathbf{x}}_i} \min_{\mathbf{s}_j \in \bar{\mathbf{x}}_j} \left\|\mathbf{s}_i - \mathbf{s}_j\right\|_2 \tag{4}$$

Given the collection of m lines, the symmetric distance matrix $\mathbf{D} \in \mathbb{R}^{m\times m}$ stores all pairwise distances. For clustering, we apply agglomerative hierarchical clustering using ALGLIB [Boc23].

Linkage. During clustering, sets of lines are merged and the distance matrix is updated accordingly. How this is done is determined by the linkage method. In the supplemental material, we discuss and demonstrate two different linkage methods, i.e., single linkage and complete linkage. We conclude that the complete linkage served us well and is the recommended default choice.

Representative Selection. Once the clusters are formed, each cluster contains a possibly varying number of lines from which a representative needs to be chosen as stand-in for the other lines in the cluster [KFW16]. In the supplemental material, we discuss and compare three different representative selection methods, including the mean line, the line with least distance to all others, and the most important line. Empirically, we found that the latter is most aligned with the goals of the opacity optimization and we thus recommend this as the default choice.

3.3. Opacity Optimization

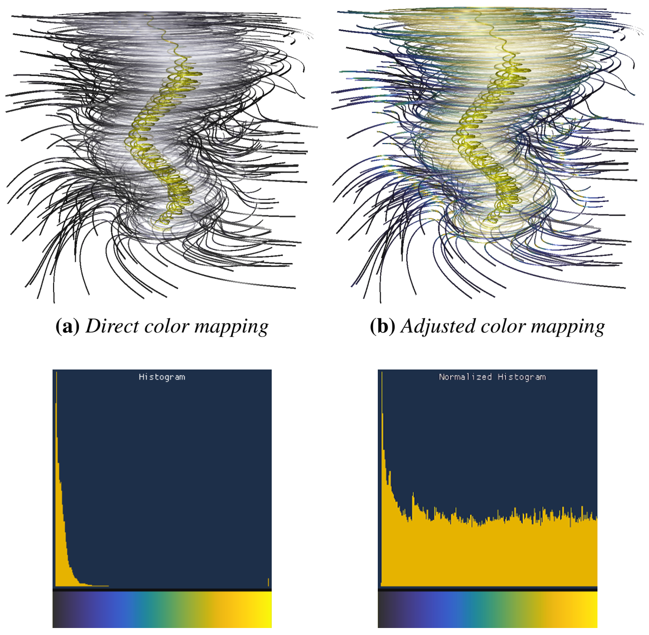
In the third step of the pipeline, the transparency is calculated for all line segments using decoupled opacity optimization [GTG17]. The aim is to maximize the visibility of important lines in screen space by adjusting opacity values of rasterized fragments such that occlusion and cluttering of important structures is reduced. See Fig. 1 (b) and (c) for an example. The algorithm is conceptually organized into three steps. First, each input line is divided into a fixed number of line segments, which allows for an optimization independent of the underlying number of vertices per line. Second, all line geometry is rasterized and the fragments that are visible in a pixel are recorded in fragment linked lists [YHGT10]. Third, the fragments are sorted and the sum of importance values in calculated in front and behind each fragment. From this, the transparency values $\alpha_i \in [0,1]$ are calculated per fragment i by minimizing the following per-fragment energy, cf. [BRGG20]:

$$E_i(\alpha_i) = \frac{1}{2}(\alpha_i - 1)^2 + \alpha_i^2 (1 - g_i)^{2\lambda} \left(\frac{q}{2} \sum_{j=1}^{i-1} g_j^2 + \frac{r}{2} \sum_{j=i+1}^{n} g_j^2 \right) \quad (5)$$

The first term is a regularizer that generally encourages high opacity values. The q-term penalizes the occlusion of important segments by lesser important ones. The r-term fades out unimportant segments behind more important ones. The parameters q, r, λ were introduced to control the extent by which importance variations influence the opacity optimization. The minimization has a closed-form solution, as was shown by Ament et al. [AZD17]. As this optimization algorithm alone leads to discontinuous opacity values at line crossings, Laplacian smoothing is applied across the line segments to obtain smooth transparency transitions [GTG17].

3.4. Color Optimization

In the final stage of the pipeline, all lines are colored such that the color space is optimally utilized. Each line vertex has a coloration value $c(t) \in [0,1]$, which is used to sample color values from a selected colormap $T(c) : [0,1] \to [0,1]^3$. However, using the coloration values directly for color mapping may lead to overused and underutilized color regions depending on the distribution of the scalar values, see Fig. 4. Furthermore, when the camera moves, the visible geometry changes due to the opacity optimization leading again to an uneven colormap usage. In value ranges that are not fixed by the user, we first redistribute the coloration values based on the visibility of line segments before applying the color mapping.



(a) *Direct color mapping* **(b)** *Adjusted color mapping*

(c) *Original coloration distribution* **(d)** *Histogram equalized distribution*

Figure 4: *Visualization of the effect of histogram equalization on color mapping. The* TORNADO *dataset is displayed and curvature is used for the coloration value $s(t)$. (a): Directly applying the color map does not convey that values near the core are elevated compared to far away regions. (b): Color mapping after redistributing the colorization values shows the difference. (c): The distribution of visible curvature values is plotted. (d): The distribution of curvature values is plotted after applying histogram equalization.*

Histogram Calculation. The opacity optimization delivers a front-to-back sorted list of fragments $\mathcal{F}_p$ per pixel $p \in \mathcal{P}$ in the set of all pixels $\mathcal{P}$. Let $c_{k,p} \in [0,1]$ be the colorization scalar value of the kth fragment of pixel p and let $\alpha_{k,p}$ be its opacity value as calculated by the opacity optimization. In front-to-back alpha blending, the total contribution of a fragment k to the final pixel color is given by the product of its transmittance $T_{k,p} = \prod_{i=1}^{k-1}(1 - \alpha_{i,p})$ and its own alpha value $\alpha_{k,p}$. To assess how well the color ranges of the color map are utilized, we form a visibility-weighted histogram over all pixels $\mathcal{P}$. Let N be the total number of histogram bins, then the histogram entry H_b of bin $b \in \{1, \ldots, N\}$ is:

$$H_b = \sum_{p \in \mathcal{P}} \sum_{\substack{k=1\ldots|\mathcal{F}_p|, \\ c_{k,p} \in [\frac{b-1}{N}, \frac{b}{N}]}} T_{k,p} \cdot \alpha_{k,p} \quad (6)$$

The first sum loops over all pixels and the second sum loops per pixel over all the fragments whose colorization scalar is within the value range of the bin. The resulting histogram for an unoptimized colorization is shown in Fig. 4(c). Note that the majority of the color map is unused, since most fragments are either in the dark blue range or are saturated at bright yellow. To make good use of the color map, we want this histogram to become more uniform.

Histogram Equalization. Histogram equalization [GW06] takes the discrete histogram H_b with its bins b and first turns it into a probability density $p : [0,1] \to [0,1]$ via normalization and afterwards integrates it to arrive at the cumulative density $F(c)$:

$$p(c) = \frac{H_{\lfloor c \cdot N \rfloor + 1}}{\sum_b H_b} \qquad F(c) = \int_0^c p(c')\,\mathrm{d}c' \tag{7}$$

The cumulative density $F(c)$ of the discrete histogram is calculated by computing the prefix sum followed by the division of all elements by the largest prefix sum value. An equalization by applying $F(c)$ to the colorization scalar is shown in Fig. 4(d). Note that the bins are not perfectly equal if duplicate values occur in scalar $c(t)$.

Exponential Decay. During camera navigation, both the opacity and the view-dependent color mapping are changing. Directly using the newly calculated transfer function $F(c)$ results in noticeable temporal discontinuities in the color mapping, as geometries emerge or vanish from the view. To ensure continuous color transitions, we add a blending between the currently used transfer function and the newly calculated one. For this purpose, we utilize exponential decay of some quantity q over time τ:

$$q(\tau) = q_0 \cdot e^{-\lambda \cdot \tau} \tag{8}$$

where q_0 is the initial amount and $\lambda > 0$ is the decay rate. The decay rate can be expressed via the half-life $\tau_{1/2} = \frac{\ln(2)}{\lambda}$, which characterizes the point in time at which the quantity has halved. Inserting the half-life into Eq. (8) gives the following formula:

$$q(\tau) = q_0 \cdot e^{-\frac{\ln(2)}{\tau_{1/2}}\tau} = q_0 \cdot 2^{-\frac{\tau}{\tau_{1/2}}} \tag{9}$$

With this, we linearly blend between the current discrete CDF value f_b and its target value $\hat{f}_b$ to produce a smooth transitory value f_b':

$$f_b' = f_b \cdot (1-\alpha) + \hat{f}_b \cdot \alpha \quad \text{with} \quad \alpha = 1 - 2^{-\frac{\Delta\tau}{\tau_{1/2}}} \tag{10}$$

where $\Delta\tau$ is the time that has elapsed since the last draw call.

4. Implementation Details

In the following, we explain how our algorithm is integrated into the decoupled opacity optimization [GTG17] pipeline.

1. Construct a fragment linked list [YHGT10] per pixel containing the depth and importance of each rasterized fragment. To speed this up, this is done at half resolution.
2. For each pixel, sort the fragment linked list from front-to-back.
3. Calculate the alpha value for each fragment via the closed-form formula given in [GTG17].
4. Smooth the opacity values in object space.
5. Apply a fading to obtain temporally smooth results.
6. For rendering, construct fragment linked list at full resolution, containing depth, alpha and coloration values of all fragments.
7. Sort the fragment lists for each pixel from front-to-back.
8. Calculate the histogram of coloration values given their visibility by rasterization and additive blending.
9. Compute the cumulative density function via a parallel prefix sum [HS86] in a compute shader.
10. Apply exponential decay to the cumulative density function (CDF) to ensure smooth color transitions.
11. Render the final image to screen using the sorted fragment linked list and the color mapping by the CDF.

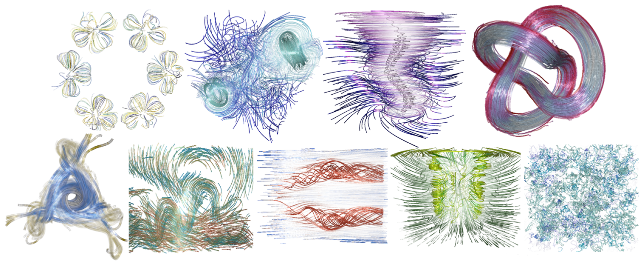


Figure 5: *All datasets used in this paper, named in order of occurrence:* BENZENE, BORROMEAN, TORNADO, TREFOIL *(timesteps 10 and 140),* ECMWF, DELTA WING, HELI *and* CTBL.

Compared to [GTG17], now step 6 includes the storage of the coloration values, and the steps 8-10 are added. Step 11 includes the color mapping. The rest is unchanged and follows the open source implementation of [GTG17].

5. Results

For evaluation, we used the datasets shown in Fig. 5, applying the perceptually uniform color maps of Kovesi [Kov15]. In the following, we study the effect of the number of bins N and of the lifting weights λ_g, λ_c. Further, we demonstrate the exponential decay, and provide a performance analysis of the visualization pipeline. For further experiments on the options for cluster linkage, representative selection, and the effect of bin size and cluster size on the performance, we refer to the supplemental material.

5.1. Number of Bins

First, we analyze the influence of the number of bins N on the transfer function and the final image itself. We extracted streamlines in the BORROMEAN dataset and compare the results of color mapping for a variety of bin sizes in Fig. 6. It is desirable to determine a reasonable default value for the number of bins. We increased the number of bins by powers of two and calculated the SSIM [WBSS04] between two subsequent images to find a suitable number of bins for which a further increase no longer makes a difference. We automatically terminate the increase when the SSIM is above 0.999, which occurred here for $N = 256$, which we likewise used for the other data sets.

5.2. Parameter Study of Lifting Weights

For the line clustering, we lifted the 3D curves into 5D by appending the importance $g(t)$ and the coloration scalar $c(t)$ with customizable weights λ_g and λ_c, cf. Eq. (2). In Fig. 7, we demonstrate the results for different parameter choices in the BORROMEAN data set. Line length was selected as importance, while vorticity was

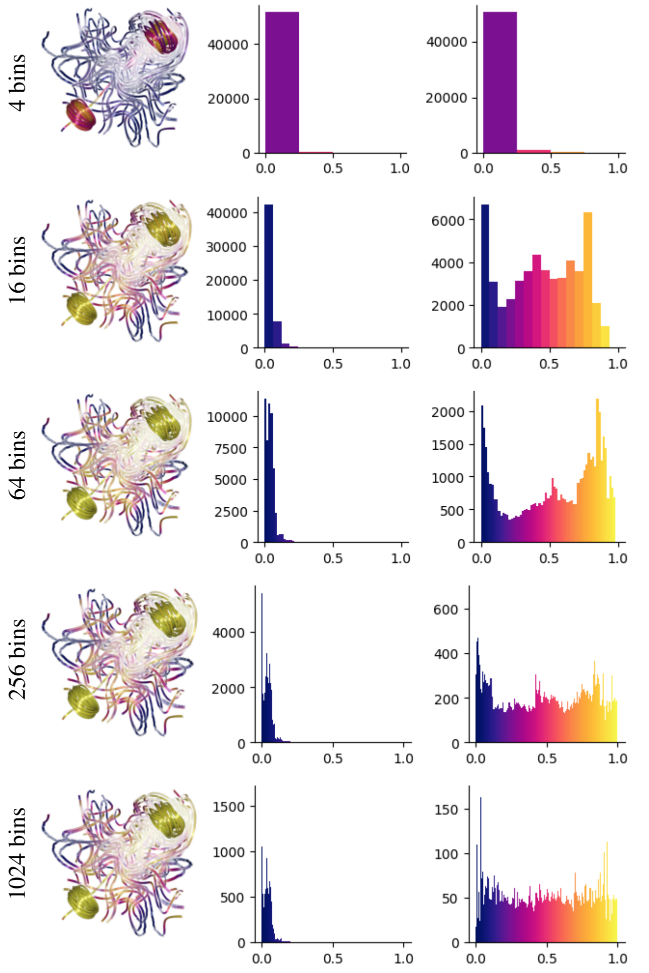


Figure 6: *Variations of the bin size N. In the second column, we plot the histogram of the initial coloration values. In the third column, the coloration values are plotted after applying the histogram equalization. Increasing the number of bins results in a more uniform distribution of values.*

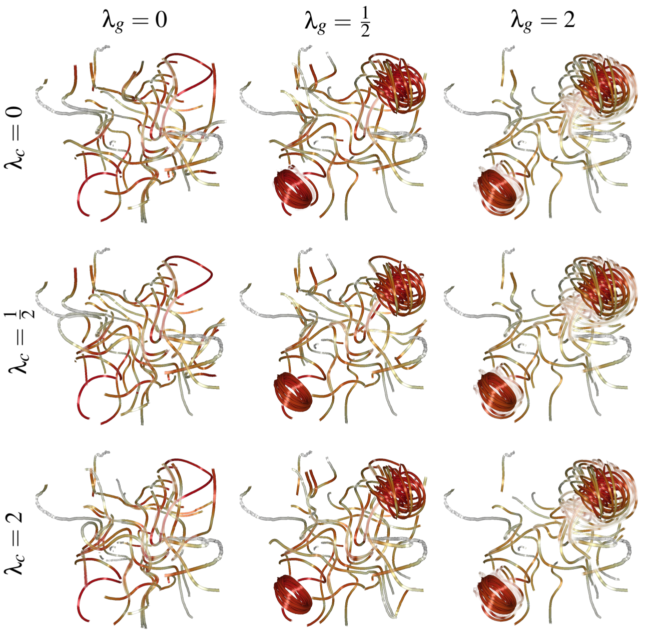


Figure 7: *Visualizations of the clusters created by varying the weighting factors for the importance λ_g and coloration values λ_c.*

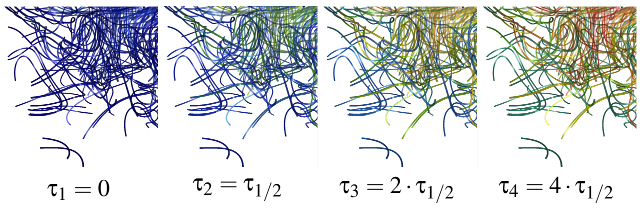


Figure 8: *Visualization of temporal color transition due to exponential decay. At τ_1, the camera jumps to the position shown on the left. With exponential decay, the currently used CDF smoothly approaches the target values. At time τ_4, the current transfer function has reached the target. In this example, $\tau_{1/2} = 1$, meaning the color mapping fully converged after about four seconds.*

picked as the coloration value. All lines were clustered into 25 clusters using complete linkage, showing the *least distance* representative. Setting the value of the importance scaling factor λ_g above 0.5 leads to the appearance of the characteristic rings. Varying the value of λ_c did not make a noticeable difference in this experiment.

5.3. Exponential Decay

The color mapping is dynamically adjusted based on the currently visible line geometry. Due to the interactivity, the visibility of lines can change rapidly through minimal changes of the camera position or viewing direction. This causes sudden changes in the transfer function, as well. To ensure smooth and continuous temporal transitions for the color mapping, we apply an exponential decay. Subsequent frames of an exponential decay are presented in Fig. 8.

We empirically found a half-life of $\tau_{1/2} = 0.8$ to be sufficient to create smooth color transitions without being too slow.

5.4. Performance

In this section, we measured the computation time (in milliseconds) for each rendering stage, as introduced in Section 4 using an AMD Ryzen 9 5900X and an NVIDIA GeForce RTX 3070 TI. All datasets are clustered to have 1,000 lines and the line widths are adjusted to a sensible value, depending on the spatial extent of each data set. The coloration value is set to the line length and the opacity optimization parameters are set to $q = 80, r = 80, \lambda = 1$ for all datasets. Lastly, the most important representative method is used and the camera is positioned such that it captures all lines of a dataset. We found that across all datasets the most expensive step is the creation of the fragment linked lists, sorting the lists, and computing the histogram. At the cost of small visual errors, the linked

Stage	Benz.	Borro.	Delta	ECM.	Trefoil	CTBL
Create list (LR)	0.26	2.08	0.63	1.05	2.84	3.52
Sort list (LR)	87.54	13.61	1.35	1.11	23.15	188.78
Comp. alpha	1.34	2.22	0.31	0.15	3.09	2.18
Smoothing	0.14	0.03	0.08	0.03	0.03	0.02
Fade alpha	0.01	0.01	0.01	0.05	0.03	0.08
Create list	0.21	1.52	0.62	1.02	2.55	2.00
Sort list	28.84	8.83	1.62	0.47	25.42	109.82
Histog.	0.42	12.66	2.33	0.43	12.96	3.58
Histog. CDF	0.02	0.06	0.11	0.02	0.12	0.01
Exp. decay	0.01	0.01	0.01	0.01	0.01	0.01
Render image	0.30	0.69	0.21	0.09	1.24	0.98
Total time	122.02	43.87	10.00	8.00	73.55	311.04

Table 1: *Compute time in ms per rendering stage, see Section 4.*

list construction can be avoided entirely using order-independent transmittance approximations [BRGG20, ZRPD20], which is orthogonal to the colorization done in this paper. The other pipeline steps have a negligible impact on the frame time. The BENZENE and CTBL datasets have the longest fragment linked list creation and sorting time, which is due to the high overdraw in those scenes. In the case of BENZENE, all extracted streamlines are grouped into similar and small areas while most of the domain is empty. Consequently, a high number of lines passes through individual pixels, which leads to long linked lists. In CTBL, the whole domain is densely covered with streamlines, which cover the entire viewport in our experiments, as shown in Fig. 5 (bottom right). For this reason, there are few empty fragment linked lists and higher overdraw. The exponential decay is unaffected by the dataset choice. This is to be expected, since the compute shader solely depends on the number of bins in the histogram.

5.5. Limitations

The color adjustment is only suitable for emphasizing relative differences (i.e., the order of values). The view-dependent adjustment loses the absolute expression of values (i.e., the magnitude of values). A shared limitation of all transparency-based methods is that excessive use of transparency inhibits the depth perception. In the future, we would like to include global illumination cues that can help to improve the perception [KRW18]. At the moment, our clustering is view-independent. Kanzler et al. [KFW16, KW18] explored view-dependent clustering, which could be utilized instead.

6. Conclusions

In this paper, we described an interactive visualization of a vector field together with a scalar field. While the vector field is visualized by streamlines, the scalar field is encoded by color. We utilized and extended the decoupled opacity optimization [GTG17] in two ways. First, we performed an agglomerative hierarchical clustering to lower the number of lines. Second, we calculated visibility histograms [CM10] using the transmittance of individual fragements. The histogram was equalized to uniformly distribute the used color map range in the final image. In the future, we would like to improve the rendering of the lines, for example by including global illumination effects [KRW18, GG21]. Further, the clustering could be optimized to further maximize the scalar value range, which would couple the two approaches. Lastly, the influence of clustering and binning parameters on different views could be studied.

Acknowledgements. We thank Hans-Christian Hege for the BENZENE data set. The BORROMEAN and TREFOIL were provided by [CB11]. The tornado was made available by Roger Crawfis. The ECMWF reanalysis is by Dee et al. [DUS*11]. The DELTA WING flow was provided by Markus Rütten. The CTBL is by Stevens et al. [Ste10] and the HELI is by Yu et al. [YTvdW*02]. This work was partially supported by DFG grant no. GU 945/3-1.

References

[AZD17] AMENT M., ZIRR T., DACHSBACHER C.: Extinction-optimized volume illumination. *IEEE Transactions on Visualization and Computer Graphics 23*, 7 (2017), 1767–1781. 2, 4

[Boc23] BOCHKANOV S.: Alglib 4.01.0 for C++. www.alglib.net, 2023. 3

[BRGG20] BAEZA ROJO I., GROSS M., GÜNTHER T.: Fourier opacity optimization for scalable exploration. *IEEE Transactions on Visualization and Computer Graphics 26*, 11 (2020), 3204–3216. 2, 4, 7

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

[CM10] CORREA C. D., MA K.-L.: Visibility histograms and visibility-driven transfer functions. *IEEE Transactions on Visualization and Computer Graphics 17*, 2 (2010), 192–204. 2, 3, 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. 3

[DUS*11] DEE D. P., UPPALA S. M., SIMMONS A. J., BERRISFORD P., POLI P., KOBAYASHI S., ANDRAE U., BALMASEDA M., BALSAMO G., BAUER D. P., ET AL.: The era-interim reanalysis: Configuration and performance of the data assimilation system. *Quarterly Journal of the royal meteorological society 137*, 656 (2011), 553–597. 7

[EBRI09] EVERTS M. H., BEKKER H., ROERDINK J. B., ISENBERG T.: Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1299–1306. 3

[EHS13] EICHELBAUM S., HLAWITSCHKA M., SCHEUERMANN G.: LineAO - improved three-dimensional line rendering. *IEEE Trans. on Visualization and Computer Graphics 19*, 3 (2013), 433–445. 3

[FTC20] FENGLIN TIAN L. C., CHEN G.: Transfer function-based 2d/3d interactive spatiotemporal visualizations of mesoscale eddies. *International Journal of Digital Earth 13*, 5 (2020), 546–566. 3

[FTG24] FRIEDERICI A., THEISEL H., GÜNTHER T.: Trajectory vorticity - computation and visualization of rotational trajectory behavior in an objective way. *IEEE Transactions on Visualization and Computer Graphics* (2024), to appear. 3

[GG21] GROSS D., GUMHOLD S.: Advanced rendering of line data with ambient occlusion and transparency. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 614–624. 3, 7

[GRT13] GÜNTHER T., RÖSSL C., THEISEL H.: Opacity optimization for 3D line fields. *ACM Trans. Graph. 32*, 4 (jul 2013). 1, 2, 3

[GRT14] GÜNTHER T., RÖSSL C., THEISEL H.: Hierarchical opacity optimization for sets of 3D line fields. *Computer Graphics Forum (Eurographics) 33*, 2 (2014), 507–516. doi:10.1111/cgf.12336. 2

[GSME*14] GÜNTHER T., SCHULZE M., MARTINEZ ESTURO J., RÖSSL C., THEISEL H.: Opacity optimization for surfaces. *Computer Graphics Forum (Eurographics Conference on Visualization) 33*, 3 (2014), 11–20. doi:10.1111/cgf.12357. 2

[GTG17] GÜNTHER T., THEISEL H., GROSS M.: Decoupled opacity optimization for points, lines and surfaces. *Computer Graphics Forum (Eurographics) 36*, 2 (2017), 153–162. doi:10.1111/cgf.13115. 1, 2, 4, 5, 7

[GW06] GONZALEZ R. C., WOODS R. E.: *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., USA, 2006. 5

[HG24] HIMMLER P., GÜNTHER T.: Transmittance-based extinction and viewpoint optimization. *Computer Graphics Forum (Eurographics Conference on Visualization) 43*, 3 (2024). 2

[HS86] HILLIS W. D., STEELE G. L.: Data parallel algorithms. *Commun. ACM 29*, 12 (dec 1986), 1170–1183. 5

[HTW20] HAN J., TAO J., WANG C.: FlowNet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Transactions on Visualization and Computer Graphics 26*, 4 (2020), 1732–1744. 2

[HWU*19] HAN M., WALD I., USHER W., WU Q., WANG F., PASCUCCI V., HANSEN C. D., JOHNSON C. R.: Ray tracing generalized tube primitives: Method and applications. *Computer Graphics Forum 38*, 3 (2019), 467–478. 3

[Jai10] JAIN A. K.: Data clustering: 50 years beyond k-means. *Pattern recognition letters 31*, 8 (2010), 651–666. 2

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. *Visualization in scientific computing 97* (1997), 43–55. 2

[KFW16] KANZLER M., FERSTL F., WESTERMANN R.: Line density control in screen-space via balanced line hierarchies. *Computers & Graphics 61* (2016), 29–39. 2, 4, 7

[KNM*20] KERN M., NEUHAUSER C., MAACK T., HAN M., USHER W., WESTERMANN R.: A comparison of rendering techniques for 3d line sets with transparency. *IEEE Transactions on Visualization and Computer Graphics 27*, 8 (2020), 3361–3376. 3

[Kov15] KOVESI P.: Good colour maps: How to design them. *ArXiv abs/1509.03700* (2015). 5

[KRW18] KANZLER M., RAUTENHAUS M., WESTERMANN R.: A voxel-based rendering pipeline for large 3D line sets. *IEEE Trans. on Visualization and Computer Graphics 25*, 7 (2018), 2378–2391. 3, 7

[KW18] KANZLER M., WESTERMANN R.: Interactive visual exploration of line clusters. In *Proceedings of the Conference on Vision, Modeling, and Visualization* (2018), pp. 155–163. 2, 7

[LMG06] LIU Z., MOORHEAD R., GRONER J.: An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 965–972. 2

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

[MAD05] MEBARKI A., ALLIEZ P., DEVILLERS O.: Farthest point seeding for efficient placement of streamlines. In *VIS 05. IEEE Visualization, 2005.* (2005), IEEE, pp. 479–486. 2

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

[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 (2010), 1807–1829. 1

[MPSS05] MALLO O., PEIKERT R., SIGG C., SADLO F.: Illuminated lines revisited. In *VIS 05. IEEE Visualization, 2005.* (2005), IEEE, pp. 19–26. 3

[MTHG03] MATTAUSCH O., THEUSSL T., HAUSER H., GRÖLLER E.: Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In *Proceedings of the 19th spring conference on Computer graphics* (2003), pp. 213–222. 2

[MWS13] MA J., WANG C., SHENE C.-K.: Coherent view-dependent streamline selection for importance-driven flow visualization. In *Visualization and Data Analysis* (2013), vol. 8654, SPIE, pp. 71–85. 2

[NWKW22] NEUHAUSER C., WANG J., KERN M., WESTERMANN R.: Efficient high-quality rendering of ribbons and twisted lines. In *Vision, Modeling & Visualization* (2022), pp. 135–143. 3

[OLK*14] OELTZE S., LEHMANN D. J., KUHN A., JANIGA G., THEISEL H., PREIM B.: Blood flow clustering and applications in virtual stenting of intracranial aneurysms. *IEEE Transactions on Visualization and Computer Graphics, 2014 20*, 5 (2014), 686 – 701. 2, 3

[RT12] RÖSSL C., THEISEL H.: Streamline embedding for 3d vector field exploration. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 407–420. 2

[SBGC20] SANE S., BUJACK R., GARTH C., CHILDS H.: A survey of seed placement and streamline selection techniques. *Computer Graphics Forum 39*, 3 (2020), 785–809. doi:10.1111/cgf.14036. 1, 2

[SGS05] STOLL C., GUMHOLD S., SEIDEL H.-P.: Visualization with stylized line primitives. In *VIS 05. IEEE Visualization, 2005.* (2005), IEEE, pp. 695–702. 3

[SHH*07] SCHLEMMER M., HOTZ I., HAMANN B., MORR F., HAGEN H.: Priority streamlines: A context-based visualization of flow fields. In *EuroVis* (2007), pp. 227–234. 2

[Ste10] STEVENS B.: Introduction to UCLA-LES, 2010. 7

[TB96] TURK G., BANKS D.: Image-guided streamline placement. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 453–460. 2

[VKP00] VERMA V., KAO D., PANG A.: A flow-guided streamline seeding strategy. In *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)* (2000), IEEE, pp. 163–170. 2

[WBSS04] WANG Z., BOVIK A., SHEIKH H., SIMONCELLI E.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612. 5

[WNW*22] WANG J., NEUHAUSER C., WU J., GAO X., WESTERMANN R.: 3D-TSV: The 3D trajectory-based stress visualizer. *Advances in Engineering Software 170* (2022), 103144. 3

[WWLM*19] WALDIN N., WALDNER M., LE MUZIC M., GRÖLLER E., GOODSELL D. S., AUTIN L., OLSON A. J., VIOLA I.: Cuttlefish: Color mapping for dynamic multi-scale visualizations. *Computer Graphics Forum 38*, 6 (2019), 150–164. 2, 3

[YHGT10] YANG J. C., HENSLEY J., GRÜN H., THIBIEROZ N.: Real-time concurrent linked list construction on the GPU. *Computer Graphics Forum 29*, 4 (2010), 1297–1304. 4, 5

[YTvdW*02] YU Y. H., TUNG C., VAN DER WALL B., PAUSDER H.-J., BURLEY C., BROOKS T., BEAUMIER P., DELRIEUX Y., MERCKER E., PENGEL K.: The HART-II test. rotor wakes and aeroacoustics with higher-harmonic pitch control (HHC) inputs - the joint German/French/Dutch/U.S. project. *Annual Forum Proceedings-American Helicopter Society 58*, 2 (2002), 1984–1994. 7

[YWSC11] YU H., WANG C., SHENE C.-K., CHEN J. H.: Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2011), 1353–1367. 2

[ZHT06] ZHANG Z., HUANG K., TAN T.: Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *18th International Conference on Pattern Recognition (ICPR'06)* (2006), vol. 3, IEEE, pp. 1135–1138. doi:10.1109/ICPR.2006.392. 2

[ZRPD20] ZEIDAN M., RAPP T., PETERS C., DACHSBACHER C.: Moment-based opacity optimization. In *Eurographics Symposium on Parallel Graphics and Visualization* (2020), The Eurographics Association. doi:10.2312/pgv.20201072. 2, 7

[ZSH96] ZOCKLER M., STALLING D., HEGE H.-C.: Interactive visualization of 3D-vector fields using illuminated stream lines. In *Proceedings of Seventh Annual IEEE Visualization'96* (1996), IEEE, pp. 107–113. 3