Automatic Seeding of Multiple Stream-Surfaces for Local Flow Investigation

Andrea Brambilla

Abstract

Integral surfaces, such as stream- and path-surfaces, are highly effective when it comes to the exploration and analysis of the long-term behavior of three-dimensional flows. However, specifying the seeding curves that lead to an expressive set of integral surfaces is a challenging and cumbersome task. In this paper, we propose an algorithm for automatically seeding multiple stream-surfaces around a user-specified point of interest. The process is guided by a streamline similarity measure. Within the resulting integral surfaces, adjacent streamlines are as similar as possible to each other. Conversely, we aim at conveying different aspects of the flow behavior with each surface. This is achieved by maximizing the dissimilarity between streamlines from different stream-surfaces. The capabilities of our technique are demonstrated on a number of application cases. We compare our expectations with the results of an informal survey. We report from our detailed exchange with a domain expert concerning the expressiveness and usefulness of our approach. A thorough analysis of the few parameters involved is provided.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation, Display algorithms

1. Introduction

Air and water currents, electricity and dynamical systems are common examples of flow phenomena. They are ubiquitous in our world, and the ability to analyze and understand them can have a deep impact on our daily life. A well established method for investigating flow phenomena is represented by *streamlines*. A streamline is a curve that starts at a *seeding point* in the spatial domain and is everywhere tangent to the flow vectors. In the case of time-independent (steady) flows, it corresponds to the trajectory of a massless particle released in the flow at the seeding point.

Even when we are interested in just a limited portion of the domain, using a single streamline may lead to incomplete information. Therefore, several streamlines are often seeded and analyzed together. In the special case that all the seeding points lie on a $1D$ curve (the *seeding curve*), the streamlines can be merged into a single *stream-surface*. Even though stream-surfaces can be more expressive, streamlines are still the primary choice in many practical contexts. The reasons are several: (1) Computing a stream-surface is more complex and more expensive than computing multiple streamlines. (2) Specifying a set of uniformly distributed seeding points is usually easier than defining a seeding curve in $3D$. (3) Finding a seeding curve that would lead to an expressive and easy to understand stream-surface is highly challenging. These three issues are amplified when multiple stream-surfaces have to be computed. Moreover, in order to exploit the available computational power at best, we should avoid computing stream-surfaces that convey the same aspect of the phenomenon of interest.

In this paper, we propose a seeding algorithm for multiple stream-surfaces that explicitly addresses issues (2) and (3). We focus on the case of local flow investigation, i.e., we assume the user is interested in seeding surfaces around a specific point of interest in the spatial domain. Our first goal is to provide a system which is simple and easy-to-use. Therefore, we ask the user to specify only the point of interest and a few other intuitive parameters. Our technique automatically constructs the seeding curves in a neighborhood of the selected location. Our second goal is to compute stream-surfaces that are easy to understand and that express the flow behavior at best. To this end, we construct the seeding curves according to the following guidelines:

(a) Each surface should capture a different aspect of the flow behavior (avoid redundancy).
(b) Each surface should capture only one aspect of the flow behavior (easy interpretation).

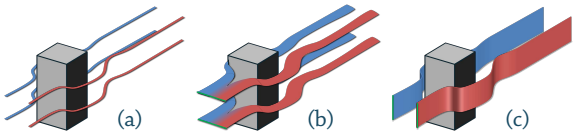


Figure 1: *(a) Streamlines of a flow around an obstacle. (b) Each of these surfaces captures the bi-modal behavior of the flow. They have a complex shape and there is no advantage in using two surfaces. (c) Each of these surfaces captures a single aspect of the flow behavior. The conveyed information is the same as in (b), but they are easier to understand.*

A simple example is given in Figure 1. As a matter of fact, these two guidelines represent the basis of several well-established techniques. For instance, *Vector Field Topology* [HH91] partitions the flow domain into regions of coherent behavior. Each partition isolates a specific flow behavior (b), which is different from the flow behavior in another partition (a). Clustering, irrespective of the entities being clustered, is another notable example. The inter-cluster similarity is minimized (a) while the intra-cluster similarity is maximized (b). In flow visualization, similarity measures for streamlines are a common way to characterize the diverse aspects of the flow behavior [MVvW05, LHS08, CYY*11, MJL*13]. We adopt this strategy as well. Given a similarity measure, a stream-surface is constructed so that adjacent streamlines within the surface are as similar as possible. At the same time, we aim at maximizing the dissimilarity between streamlines from different stream-surfaces.

In order to generate the seeding curves starting from the similarity measure, we propose a novel technique based on *multidimensional scaling* [Kru64, CC10] and *tensor analysis* (Sec. 3). We demonstrate the capabilities of our approach over a number of application scenarios, including both analytical and simulated data. (Sec. 4). Our current implementation of the algorithm exploits the computational power of both the CPU and the GPU. The computation time varies from tenths of a second up to a few seconds, allowing for the interactive exploration of flow datasets (Sec. 7). We conducted a preliminary survey (Sec. 5) and discussed our system with a collaborator from the CFD domain (Sec. 6). The ease of use and the clarity of the results were among the most appreciated qualities of our technique.

The main contributions of this paper are the following:

- We present a novel approach for automatically seeding multiple stream-surfaces in the context of local flow exploration.
- We implemented our approach in an interactive and easy-to-use system.
- We compare selected streamline similarity measures with respect to their ability to characterize the flow behavior.
- We exploit multidimensional scaling (MDS) and tensor analysis in an innovative way, in order to handle the similarity between multiple entities in a continuous space.

2. Related Work

The study of integral curves and integral surfaces is an active sub-field of flow visualization with a 30 years long history. Reviewing this large amount of literature is beyond the scope of this paper. Interested readers are referred to three recent state-of-the-art reports [MLP*10, BCP*12, ELC*12a] which thoroughly describe the field. Here we present only a selection of approaches related to our technique. Specifically, we subdivide the related work in three categories: *Streamline placement and similarity measures*, *Stream-surface construction and rendering*, *Automatic stream-surface seeding*.

Streamline placement and similarity measures. The aim of streamline placement techniques is to display a large number of streamlines while avoiding cluttering and occlusion problems. The main issue is that a uniform arrangement of seeding points does not lead to a homogeneous distribution of streamlines. A possible solution is to partition the flow field according to a given clustering algorithm. Then, an integral curve is seeded at the center of each cluster [TVW99]. However, this approach does not allow for controlling the streamline distribution explicitly. In contrast, the *evenly-spaced streamlines* technique [JL97] guarantees that streamlines are homogeneously spread across the spatial domain. A different strategy is adopted by Li et al. [LHS08], who aim at revealing the flow behavior with a minimal set of streamlines. Li and Shen [LS07] compute an uncluttered distribution of streamlines in $3D$ by evenly distributing curves in screen-space. One of the first approaches based on a similarity measure has been proposed by Chen et al. [CCK07]. They compare pairs of streamlines according to shape, direction and distance. In the end, they place a larger number of seeding points nearby highly dissimilar streamlines. Xu et al. [XLS10] compare streamlines according to an entropy-based measure. Then, integral curves considered redundant are removed. Linear entropy and angular entropy are instead used by Chen et al. [CYY*11] in order to cluster a dense set of streamlines. Streamline clustering is also widely employed in the context of Diffusion Tensor Imaging. An overview of the topic has been presented by Moberts et al. [MVvW05]. McLoughlin et al. [MJL*13] deal with the problem of distributing streamlines along a rake. For each streamline, they compute a signature based on three shape descriptors: curvature, torsion and tortuosity. The similarity is evaluated between pairs of signatures. We have implemented their similarity measure in our system. Tightly connected to our work is the approach by Rössl and Theisel [RT12]. They adopt the Hausdorff distance as a similarity measure for streamlines. A dissimilarity matrix is computed over a large set of curves and they process it with an MDS algorithm. Our technique follows approximately the same idea up to this point. Rössl and Theisel continue

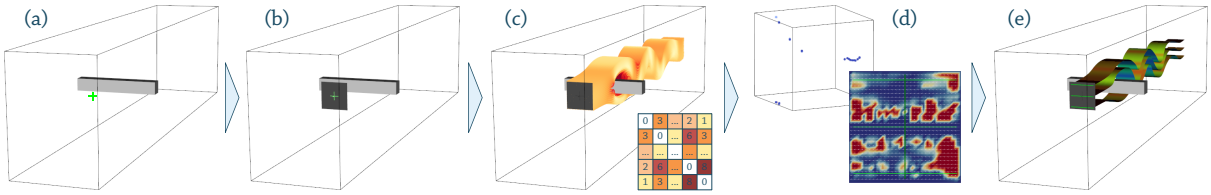


Figure 2: *Overview of our technique. (a) The user selects a point of interest. (b) A seeding structure is constructed around the selected point. (c) Streamlines are seeded from the seeding structure, and the dissimilarity between each pair of streamlines is computed. (d) The dissimilarity matrix is processed in order to determine the desired seeding curves (more details in Section 3.3) (e) Stream-surfaces are integrated from the computed seeding curves.*

by clustering the curves in the scaled space. In contrast, we process the scaled coordinates further in order to better characterize the changes in similarity in the spatial domain.

Stream-surface construction and rendering. The computation of stream-surfaces has been investigated for the first time by Hultquist [Hul92] in 1992. His algorithm is based on the idea of propagating the seeding line in the flow field (*advancing fronts*) and triangulating the positions at each step. Stalling [Sta99] improves this technique by treating critical points separately. More recently, Garth et al. [GKT*08] adopt a time-line approximation scheme as an alternative to the advancing fronts. Schultze et al. [SGRT12] instead force the front-line to be perpendicular to the flow, therefore producing high quality meshes. Rendering approaches specifically designed for integral surfaces have been investigated as well. Löffelmann et al. [LMGP97] propose the *stream arrows*, a technique for conveying the direction of the flow on the surface while reducing self-occlusion. Another possibility is represented by texture advection techniques. For instance, Van Wijk [vW03] describes how to apply the *Image Based Flow Visualization* algorithm to curved surfaces. In recent years, Born et al. [BWF*10] suggest to improve visibility and shape perception by means of contour lines, halftoning and transparency. Visibility issues are addressed directly by Brambilla et al. [BVH12], who split stream-surfaces in a hierarchical fashion. Different strategies for controlling the transparency of the surface are presented by Hummel et al. [HGH*10] and Carnecky et al. [CFM*13].

Automatic stream-surface seeding. The automatic placement of one or multiple stream-surfaces has been addressed only in recent years. The heart of the problem is defining a suitable seeding curve, or set of seeding curves. In analogy with streamline placement, clustering the spatial domain is a viable option. A seeding curve can then be placed in every cluster. Specifically, Edmunds et al. [ELM*12] compute the seeding curves by integration over the curvature vector field, starting from the center of each cluster. Another technique by Edmunds et al. [EML*11] evaluates the flow direction close to the borders of the domain, producing a scalar field over the boundary surfaces. Iso-lines of such a field are used as seeding curves. A later extension of this approach [ELC*12b] makes it possible to seed surfaces even in the inner regions of the dataset. Notice that these three techniques aim at producing a dense coverage of the spatial domain, and are therefore unsuitable for investigating the flow behavior at selected locations. Esturo et al. [ESRT13] employ a simulated annealing optimization in order to determine the single stream-surface that expresses the flow behavior at best. Their approach allows for an adequate control over the shape and the appearance of the surface, therefore it is suitable for presentation purposes. However, the location of the seeding curve cannot be directly specified. In all these approaches, one or more non-photorealistic rendering techniques are employed in order to improve shape and depth perception.

3. Automatic Surface Construction

Our technique is summarized in Figure 2. After a dataset has been loaded, the user selects a point of interest in the spatial domain. A $2D$ seeding structure of variable size and resolution is then created around the selected location. (Sec. 3.1). The next step is to compute a dense set of streamlines, one for each point of the seeding structure. The dissimilarity between each pair of streamlines is computed according to a similarity measure selected by the user (Sec. 3.2). The resulting dissimilarity matrix is processed by an MDS algorithm, which maps the streamlines to points in a scaled space. For each streamline, we compute the Jacobian of the coordinates in the scaled space. We define a *similarity tensor field* over the seeding structure, based on the Jacobian (Sec. 3.3). Finally, by computing tensor lines in the similarity tensor field, we obtain the desired seeding curves (Sec. 3.4). The user is free to modify the point of interest or any of the parameters of the system and restart the computation, producing new surfaces. In the following, all the steps of our algorithm are described in details.

3.1. Seeding

In the beginning, the user can freely select a point of interest $\mathbf{p}$ in the spatial domain $\Omega \subseteq \mathbb{R}^3$. For simplicity, we

now consider the case where only one surface has to be constructed. The extension to multiple surfaces is discussed in Section 3.5.

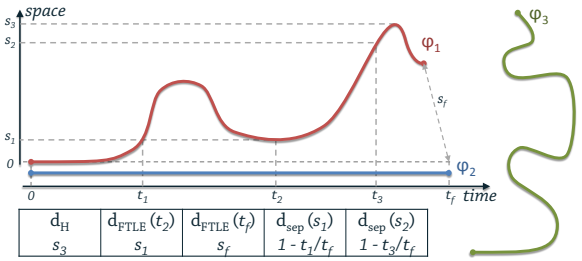
Our initial requirement is that the resulting stream-surface should pass through $\mathbf{p}$. Previous research on this topic [ELM*12, ESRT13] concluded that a desirable property for seeding curves is to be orthogonal to the flow direction. Therefore, we evaluate the velocity $\mathbf{v_p} = \mathbf{v}(\mathbf{p})$ in $\mathbf{p}$. We constrain the target seeding curve to pass through $\mathbf{p}$ and to lie in the plane orthogonal to $\mathbf{v_p}$. As a matter of fact, we do not consider the whole plane, but we take into account only a small square patch $\Psi \subset \Omega$ around $\mathbf{p}$. We refer to the square patch Ψ with the term *seeding plane*. Ψ can be naturally parameterized using two parameters $u, w \in [0,1]$. We define a *seeding structure* as the Cartesian grid obtained by subdividing Ψ in N rows and N columns, with the same constant spacing in both the u and w direction. The size of Ψ and the grid's resolution N can be controlled by the user.

3.2. Streamlines and Similarity Measures

We integrate $M = N \cdot N$ streamlines, one from each of the vertices of the seeding structure. The integration is performed with an adaptive Runge-Kutta method of 4th order. The user is free to modify the usual integration parameters, such as the desired precision or the maximal integration time. Then, we evaluate the dissimilarity between each pair of streamlines by means of a user-specified similarity measure d. The result is a symmetric distance matrix $\mathbf{D} \in \mathbb{R}_+^{M \times M}$. Specifically, let φ_i and φ_j be two of the computed streamlines. Then $\mathbf{D}_{ij} = d(\varphi_i, \varphi_j)$, where d is the adopted similarity measure. $\mathbf{D}_{ij}$ is inversely proportional to how similar φ_i and φ_j are. $\mathbf{D}_{ii} = 0$. Note that the adjacency between streamlines is taken into account at a later stage of the algorithm. Therefore d has to be invariant with respect to the location of the seeding points. This is achieved by translating each curve so that its seeding point corresponds to the origin of the coordinate system.

The user is free to choose the similarity measure that conforms to his/her requirements at best. The distribution and the shape of the resulting stream-surfaces are affected by this choice. Different similarity metrics concretize in different properties of the final surfaces. Our system is highly modular and virtually any kind of similarity measure can be easily incorporated. For demonstration purposes, we have developed four similarity measures: *Hausdorff*, *FTLE-based*, *Separation-based* and *Shape-based*. In the following, we briefly describe how they are defined and how they are expected to affect the final surfaces. Figure 3 illustrates the differences between the four measures with a simple example. Their impact in practical cases is discussed in Section 4.

Hausdorff distance. The *Hausdorff distance* is a measure defined over subsets of a metric space. Note that, in our case, streamlines are subsets of the metric space $\mathbb{R}^3$. The



d_H	$d_{FTLE}(t_2)$	$d_{FTLE}(t_f)$	$d_{sep}(s_1)$	$d_{sep}(s_2)$
s_3	s_1	s_f	$1 - t_1/t_f$	$1 - t_3/t_f$

Figure 3: *The table shows the dissimilarity values between the streamlines φ_1 and φ_2 according to the different similarity measures (with different parameters). The shape-based measure cannot be precisely evaluated in this example, but the curves' shape suggest that $d_{shape}(\varphi_1, \varphi_2)$ is by far larger than $d_{shape}(\varphi_1, \varphi_3)$.*

Hausdorff distance d_H between two streamlines φ_i and φ_j is defined as

$$d_H(\varphi_i, \varphi_j) = max\left\{d_h(\varphi_i, \varphi_j), d_h(\varphi_j, \varphi_i)\right\},$$

where

$$d_h(\varphi_i, \varphi_j) = \max_{\mathbf{x} \in \varphi_i} \min_{\mathbf{y} \in \varphi_j} \|\mathbf{x} - \mathbf{y}\|$$

and $\|\cdot\|$ is the common Euclidean norm. In other words, φ_i and φ_j are never further than $d_H(\varphi_i, \varphi_j)$ away from each other. The use of the Hausdoff distance as a streamline similarity measure is thoroughly discussed by Rössl and Theisel [RT12]. In our approach, a stream-surface obtained by adopting d_H tends to have an almost constant width, measured orthogonally with respect to the flow vectors. Consequently, it rarely crosses areas of divergent behavior, such as saddle points.

FTLE-base similarity. The *Finite Time Lyapunov Exponent* (FTLE) measures the separation rate for nearby particles over a finite period of time. It is related to the so-called *Lagrangian Coherent Structures* [Hal01], which have a significant role in the segmentation of unsteady flows. Their importance is thoroughly discussed in a recent survey [PPF*11]. We define a similarity measure d^t_{FTLE} that captures the same flow behavior as the FTLE does. We denote with $\varphi(0)$ the seeding point of the streamline φ, and with $\varphi(t)$ the point reached by integrating the flow field from $\varphi(0)$ for a time t. Then, we set

$$d^t_{FTLE}(\varphi_i, \varphi_j) = \left\|\varphi_i(t) - \varphi_j(t)\right\|.$$

The value of t is, by default, the maximal integration time used for computing the streamlines, but it can be modified by the user. Whenever we write d_{FTLE}, we assume the default parameter value is used. In analogy with the FTLE, d^t_{FTLE} does not take into account the shape of the streamlines, but only their points at time t. When d^t_{FTLE} is adopted, it is unlikely that the stream-surface will traverse regions of significant separation (up to integration time t).

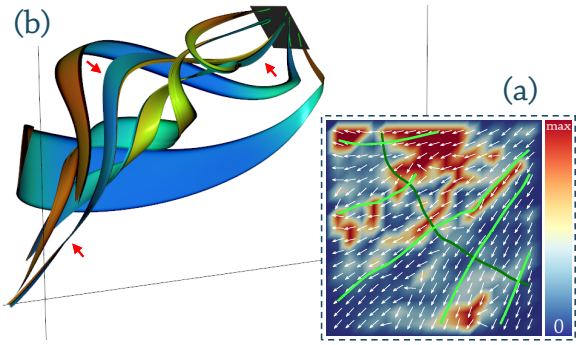


Figure 4: *Example with the basic elements of our system. Five surfaces are computed in a flow through a box. $N = 19$, d_{shape} is adopted. (a) The seeding plane Ψ colored according to the λ_{min} values. The $\mathbf{e}_{min}$ (white arrows), the seeding curves (light green lines) and the $\mathbf{e}_{max}$ tensor line (dark green line) are also shown. (b) The resulting stream-surfaces. The seeding plane (gray) and the seeding curves (light green) are displayed as well. The red arrows highlight the initial surface.*

Separation-based similarity. The separation-based similarity measure d^s_{sep} can be thought as the dual of d^t_{FTLE}. While d^t_{FTLE} takes into account the separation at time t, d^s_{sep} considers how long it takes for the curves to separate more than s, with s specified by the user. More formally, we set

$$d^s_{sep}(\varphi_i, \varphi_j) = 1 - \frac{\tilde{t}}{t_{max}},$$

where $\tilde{t}$ is the minimal time value such that $\left\|\varphi_i(\tilde{t}) - \varphi_j(\tilde{t})\right\| \geq s$, and t_{max} is the time length of the longest of the two curves (in time units). As a matter of fact, d^s_{sep} is strictly related to the *Finite Size Lyapunov Exponent* [ABC*96]. A stream-surface obtained by employing d^s_{sep} is usually composed by streamlines which stay close together at early integration times, but can significantly separate at a later stage.

Shape-based similarity. We define d_{shape} as the similarity measure introduced by McLoughlin et al. [MJL*13]. This measure explicitly takes into account shape characteristics of the curves, i.e., curvature, torsion and tortuosity. These properties are used to compute a *streamline signature* for each curve. The similarity between two streamlines is evaluated by comparing the two corresponding signatures. We refer to the original paper for a thorough description of this technique. We opted for the non-hierarchical signatures, since they offer a suitable trade-off between accuracy and performance. In contrast with the other measures, d_{shape} is very sensitive to changes in the direction of the flow along the streamlines. Therefore, the resulting stream-surface can traverse regions of even strong shear, but hardly regions that present a large variance of flow directions.

3.3. MDS and Similarity Tensor Field

The input to this stage of the pipeline is the previously computed distance matrix $\mathbf{D}_{ij} = d(\varphi_i, \varphi_j)$. We are now in need of a formal framework for investigating changes in streamline similarity with respect to the location of the seeding points. We accomplish this by applying an MDS algorithm to $\mathbf{D}$, therefore obtaining a mapping $\mathbf{f} : \{\varphi_0, \varphi_1, \ldots, \varphi_{M-1}\} \to \mathbb{R}^Q$. The MDS process guarantees that the Euclidean distance $\left\|\mathbf{f}(\varphi_i) - \mathbf{f}(\varphi_j)\right\|$ in the embedding $\mathbb{R}^Q$ is directly proportional to the dissimilarity $d(\varphi_i, \varphi_j)$. According to the analysis by Rössl and Theisel [RT12], we set $Q = 3$. Notice that each streamline φ_i has a univocal seeding point $\varphi_i(0)$ on the seeding plane Ψ. Therefore, we can define a second mapping $\mathbf{g}$ such that $\mathbf{g}(\varphi_i(0)) = \mathbf{f}(\varphi_i)$ for each $i \in \{0, 1, \ldots, M-1\}$. In order to make $\mathbf{g}$ continuous, we apply a simple Gaussian filtering, obtaining the corresponding mapping $\bar{\mathbf{g}} : \Psi \to \mathbb{R}^3$.

Recall that our goal is to construct a seeding curve such that adjacent streamlines are as similar as possible. We construct the seeding curve iteratively. We start from the initial point of interest $\mathbf{p} \in \Psi$. Our problem can be restated as determining a point $\mathbf{p} + \delta\mathbf{p}$ in an infinitesimal neighborhood of $\mathbf{p}$ such that the length of $\delta\bar{\mathbf{g}} = \bar{\mathbf{g}}(\mathbf{p} + \delta\mathbf{p}) - \bar{\mathbf{g}}(\mathbf{p})$ is *minimized*. Function $\bar{\mathbf{g}}$ is continuous, and therefore differentiable. Its Jacobian $\mathbf{J}$ is given by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \bar{g}_0}{\partial u} & \frac{\partial \bar{g}_0}{\partial w} \\ \frac{\partial \bar{g}_1}{\partial u} & \frac{\partial \bar{g}_1}{\partial w} \\ \frac{\partial \bar{g}_2}{\partial u} & \frac{\partial \bar{g}_2}{\partial w} \end{bmatrix},$$

where (u, w) is the previously defined parameterization of Ψ. It can be easily computed by central differences. We approximate $\delta\bar{\mathbf{g}}$ by its first order Taylor expansion:

$$\delta\bar{\mathbf{g}} = \bar{\mathbf{g}}(\mathbf{p}) - \bar{\mathbf{g}}(\mathbf{p} + \delta\mathbf{p}) \approx \mathbf{J}\,\delta\mathbf{p}.$$

Its Euclidean norm is given by

$$\|\delta\bar{\mathbf{g}}\| \approx \sqrt{\langle \mathbf{J}\,\delta\mathbf{p}, \mathbf{J}\,\delta\mathbf{p} \rangle} = \sqrt{\langle \delta\mathbf{p}, \mathbf{J}^T\mathbf{J}\,\delta\mathbf{p} \rangle},$$

where $\langle \cdot \rangle$ denotes the usual inner product. We define the *similarity tensor field* by computing $\mathbf{J}^T\mathbf{J}$ at each point of the seeding structure. $\mathbf{J}^T\mathbf{J} \in \mathbb{R}^{2\times 2}$ is symmetric and positive definite, so its spectral decomposition yields positive eigenvalues $\lambda_{min}, \lambda_{max}$ and orthogonal eigenvectors $\mathbf{e}_{min}, \mathbf{e}_{max} \in \mathbb{R}^2$. I turns out that $\|\delta\bar{\mathbf{g}}\|$ is minimized when $\delta\mathbf{p}$ is chosen to be parallel to $\mathbf{e}_{min}$, and its value is proportional to λ_{min}. In Figure 4a, the seeding plane Ψ is colored according to the values of λ_{min}, and arrows show the corresponding $\mathbf{e}_{min}$ vectors. It is worth pointing out that a similar argument can be found in the work by, e.g., Haller [Hal01] and Shadden et al. [SLM05] for describing the FTLE.

3.4. Seeding Curves and Surface Construction

Starting from the point of interest $\mathbf{p}$, the next point on the desired seeding curve is $\mathbf{p} + \delta\mathbf{p}$, where $\delta\mathbf{p}$ is a vector of infinitesimal length parallel to $\mathbf{e}_{min}$. This process is then re-

peated from the new point. Conceptually, we construct the seeding curve by following the $\mathbf{e}_{min}$ vectors from $\mathbf{p}$ in forward and backward direction.

The resulting curve is a *tensor line* of $\mathbf{J}^T\mathbf{J}$. There is an important difference between computing tensor lines and integrating a vector field. $\mathbf{e}_{min}$ is an eigenvector, therefore $-\mathbf{e}_{min}$ is an eigenvector as well. This is a well known issue in tensor visualization [KASH13]. We compute the desired tensor line by adopting a modified version of the interpolation algorithm by Hotz et al. [HSNHH10]. Their original algorithm assumes the domain is triangulated, while our seeding structure is a $2D$ Cartesian grid. We triangulate the square cells of our grid on the fly, during the computation of the tensor line. Out of the two possible triangulations, we choose the one that avoids degenerate points. If a degenerate point is present in both the triangulations, we follow the strategy by Delmarcelle and Hesselink [DH93] and interrupt the tensor line. Otherwise we proceed with the interpolation of the eigenvectors. If the resulting eigenvector is oriented against the previous integration direction, we flip it.

Once the seeding curve is computed (Fig. 4a, central light green curve), we employ Stalling's approach [Sta99] for constructing the related stream-surface (Fig. 4b, red arrows). The underlying integration technique is the same used for computing the streamlines (Sec. 3.2), with the same configuration of the parameters.

3.5. Multiple Stream-Surfaces

A first stream-surface, passing through the selected point $\mathbf{p}$, has been computed. Now our goal is to construct additional surfaces that capture different aspects of the flow behavior. In other words, we have to identify streamlines which are dissimilar from the ones in the first surface. Starting from $\mathbf{p}$, we look for the point $\mathbf{p}+\delta\mathbf{p}$ such that $\|\delta\bar{\mathbf{g}}\|$ is *maximized*. In analogy with our previous analysis (Sec. 3.3), this is achieved by choosing $\delta\mathbf{p}$ to be parallel to $\mathbf{e}_{max}$. We compute the tensor line along the eigenvector $\mathbf{e}_{max}$ (Fig. 4a, dark green line). Notice that streamlines seeded from adjacent points along this line would be as dissimilar as possible. Starting from $\mathbf{p}$, we uniformly distribute a user-defined number of points along the $\mathbf{e}_{max}$ tensor line. By computing the $\mathbf{e}_{min}$ tensor lines from these new points, we obtain the desired seeding curves (Fig. 4a, short light green lines). The additional surfaces are computed from these seeding curves (Fig. 4b).

3.6. Visualization

Stream-surfaces are rendered with two different colors for the front- and back-side (orange and blue) in order to ease their interpretation in presence of swirling motion (Fig. 4b). Areas of high flow curvature are shaded with a light greenish color. Phong-shading is adopted to improve depth perception and to emphasize sharp edges. View-dependent contours and silhouettes are employed to better convey shape and depth order. The seeding plane Ψ and the seeding curves are rendered next to the stream-surfaces, in gray and light green respectively.

In a separate window (Fig. 4a), we show the plane Ψ colored according to the values of λ_{min}. The seeding curves are shown in light green, while the $\mathbf{e}_{max}$ tensor line is in dark green. The white arrows represent the $\mathbf{e}_{min}$ eigenvectors, but they are usually hidden in order to avoid cluttering.

4. Demonstration

We showcase the capabilities of our technique by computing multiple stream-surfaces in four different datasets. These application scenarios are also exploited for presenting additional insights about the parameters of our algorithm. Figure 4 shows the result of our technique on a CFD simulation of a flow through a box. The inlet is on the top of the box, on the right, while the outlet is on the bottom left. The five surfaces we have computed effectively convey the vortical motion of the fluid when approaching the outlet. The rapid twisting of the two upper surfaces can indicate the presence of a vortex core nearby. The two right-most surfaces instead gives an idea of the extent of the swirling area.

Our next application case is a CFD simulation of a fluid flow around a square cylinder. We placed the point of interest right before the cylinder. The size of the seeding structure is approximately twice the height of the cylinder, and we set the resolution $N = 31$. In Figure 5a, we can see the result obtained by adopting the d_{shape} distance measure. In our opinion, this choice of surfaces seemed quite unnatural, so we investigated this case further. It turns out that the left and the right halves of the dataset have torsion values of opposite sign, probably because of the symmetric twisting motion induced by the obstacle. d_{shape} is highly sensitive to this kind of behavior, and this affects the shape of the resulting surfaces. By adopting d^s_{sep} instead (Fig. 5b) we obtain a more intuitive result. Parameter s has been set to one tenth of the diagonal of a cell of the dataset. The difference between Figures 5a and 5b may suggest that the twisting motion has only a limited extent.

Figures 5c and 5d were obtained by setting $N = 21$ and $N = 11$. All the other parameters were set as in Figure 5b. The seeding curves are slightly shifted towards the point of interest $\mathbf{p}$. This is what cause the sharp turn in the bottom stream-surface in Figure 5d. However, this is simply a consequence of the larger size of the cells in the seeding structure. The effect can be compensated by adjusting the distance of the additional surfaces with respect to $\mathbf{p}$. In general, the higher the resolution is, the better the similarity tensor field is characterized. In all the cases we investigated, adjusting the resolution never led to drastic changes in the resulting surfaces.

The *ABC flow* dataset is obtained from an analytical

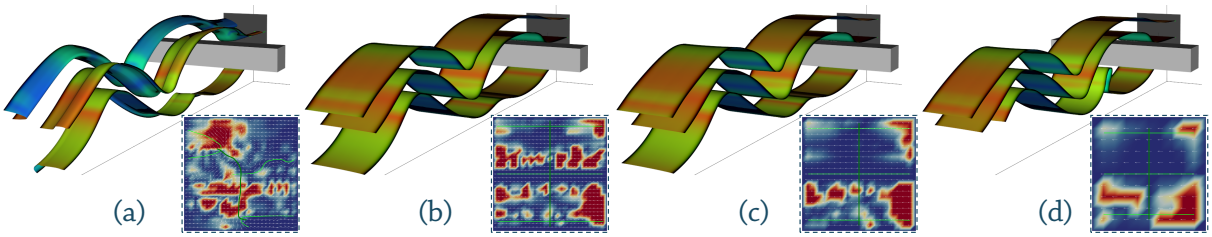


Figure 5: *Three stream-surfaces in the Square Cylinder dataset. (a) d_{shape} is heavily affected by the small changes of torsion values behind the cylinder, and this fact has a notable impact on the resulting surfaces ($N = 31$). (b) Adopting d^s_{sep}, the surfaces are less sensitive to these changes ($N = 31, s = 0.1 \cdot cell\_diagonal$). (c, d) Same as (b), but N is set to 21 and 11 respectively. Besides a slight shift of the seeding curves, no significant change can be detected.*

solution of Euler's equation for inviscid flows [DFH*86]. It is characterized by elongated vortices and large strain-dominated regions. Figure 6a shows three stream-surfaces in the ABC flow obtained with our technique. $N = 23$, d_H was employed. The left-most surface has, as expected, an almost constant width. The same is true for the one on the bottom-right (it cannot be seen from a single point of view because of its twisting behavior). The central one instead shows a significant amount of divergence and it ends in a saddle point. This is in contrast with our previous analysis of d_H (Sec. 3.2). The reason can be found by looking at the plot of the λ_{min} values. The bottom part of the central seeding curve has very high values of λ_{min}. In other words, even the most similar streamline is quite different from the current one. In order to make our tool more versatile, we give the user the opportunity to specify a threshold Λ. The computation of the $\mathbf{e}_{min}$ tensor lines is stopped whenever $\lambda_{min} > \Lambda$. This feature can be easily turned on and off by the user. The resulting surfaces (Fig. 6b) capture three different aspects of the flow behavior and present them in a clear and understandable way.

Our last application case is a simulation of a gas leak in a closed room. Three obstacles are present in the room. We place the point of interest downstream from the leak, close to the left obstacle, and we construct three surfaces. Figures 7a and 7b show the result obtained by adopting d^s_{sep} ($s = 0.2 \cdot cell\_diagonal$). We can clearly identify the vortical motion in front of the left obstacle, the laminar behavior in the center, and the flow enveloping the obstacle. As discussed in Section 3.2, d^s_{sep} can lead to surfaces with a strong divergent behavior at late integration times (Fig. 7, red arrows). Accordingly, the λ_{min} plot shows no trace of such behavior. The situation changes when d_{FTLE} is adopted (Fig. 7c). The flow under the obstacle is now isolated and depicted by a single surface. The central surface captures both the laminar motion in the center and the divergent flow enveloping the left obstacle (Fig. 7c, red arrows). This kind of behavior is unexpected when using d_{FTLE}. In analogy with the ABC flow case, the λ_{min} plot reveals that this is due to large values of the minimum eigenvalue. Figure 7d shows the result obtained by enforcing $\lambda_{min} < \Lambda$ along the seeding curve.

5. The Right Tensor Lines

Adopting the $\mathbf{e}_{min}$ tensor lines as seeding curves maximizes the similarity of adjacent streamlines. If we had selected the $\mathbf{e}_{max}$ tensor lines instead, adjacent streamlines would be maximally dissimilar. In such a case, a single stream-surface would convey multiple aspects of the flow behavior, while some information would be replicated over multiple surfaces (Fig. 1b). A trade-off can be achieved by employing the $\mathbf{e}_\alpha$ tensor lines, where the $\mathbf{e}_\alpha$ field is obtained by rotating $\mathbf{e}_{min}$

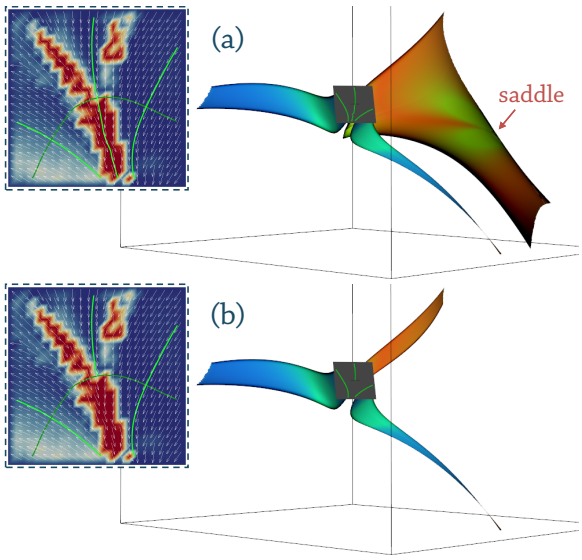


Figure 6: *Three stream-surfaces of the ABC flow ($N = 23$, d_H). (a) The central stream-surface shows elevated divergence and crosses a saddle point. Notice that the related seeding curve traverses an area with high values of λ_{min}. (b) The seeding curve is interrupted when λ_{min} becomes greater than a user-specified threshold.*

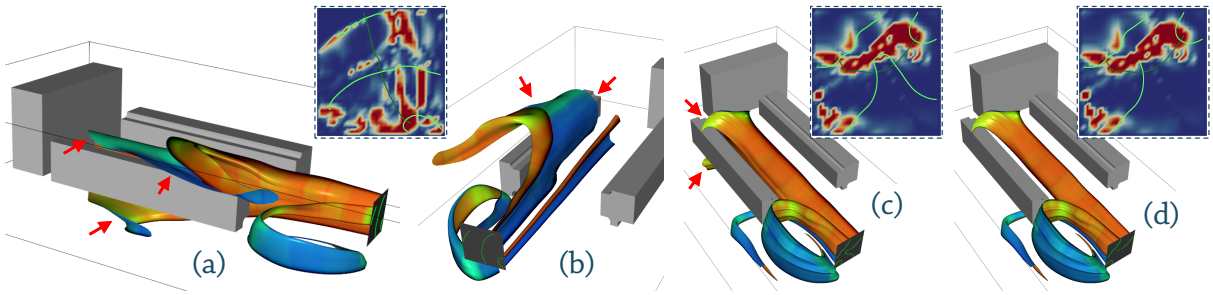


Figure 7: *Three stream-surfaces in the gas leak dataset (N = 31). (a, b) d^s_{sep} is adopted ($s = 0.2 \cdot cell\_diagonal$). d^s_{sep} allows for strong divergent behavior (red arrows). Note that this behavior is not manifested in the λ_{min} plot. (c) d_{FTLE} is adopted. The divergence around the obstacle is still present (red arrows), but it corresponds to large values of λ_{min}. (d) By setting a threshold on λ_{min}, the divergent part of the surface can be removed.*

by α degrees. Note that $\mathbf{e}_{90^\circ} = \mathbf{e}_{-90^\circ} = \mathbf{e}_{max}$. Also, in case of multiple surfaces, $\mathbf{e}_{max}$ has to be rotated accordingly. We implemented this feature in our system, letting the user specify the value of α (0° by default). In Figures 8a and 8b the eigenvectors have been rotated by, respectively, 30 and 90 degrees with respect to the case in Figure 4.

We conducted an informal survey to acquire a preliminary indication of the users' preferences. We showed the participants surfaces generated at 4 different values of α, i.e., $0^\circ, 30^\circ, 60^\circ, 90^\circ$. Participants were asked to identify: (1) Which of the 4 scenarios is the easiest to understand, and which one the most difficult to understand. (2) Which scenario is the most redundant, and which one the least redundant. (3) Which scenario conveys the largest amount of information, and which one the smallest amount. We had 10 participants, and each of them answered these questions over 4 study cases, for a total of 40 evaluations. Even though this basic investigation does not amount to a rigorous statistical analysis, we can still gain certain empirical evidence concerning the usefulness and effectiveness of our approach.

The outcome of our study is summarized in Figure 9. The clearest results concern redundancy, which seems to be directly proportional to the rotation α of the tensor lines. Scenarios with $\alpha = 0^\circ$ were selected as the easiest to understand in most of the evaluations. However, the situation here is slightly more uncertain, since the data in the "most difficult to understand" category does not show a corresponding trend. In general, these results suggest that the surfaces produced with our technique comply with our two initial guidelines, i.e., no redundancy and easy interpretation (Sec. 1). Regarding the information content, the situation is highly uncertain and no clear pattern can be identified in the data. Interestingly, $\alpha = 60^\circ$ turned out to be worse than $\alpha = 90^\circ$ in all but the "most redundant" category. It is worth pointing out that only one of the participants has a background from flow visualization. In many occasions, different participants reported that the evaluation task was challenging, since they lacked a sufficient knowledge for understanding the results fully. The uncertainty concerning the third question may be related to this fact.

6. Expert's Evaluation

In the late stages of development we had a thorough exchange with a CFD engineer from the same company that provided the gas leak dataset. The aim of this collaboration was to identify the most valuable features of our technique and to define possible directions for future improvements. The expert acknowledged the ability of our technique to capture meaningful aspects of the fluid's motion. Specifically, he pointed out that the resulting stream-surfaces are easy to understand and effective in exposing details of the flow behaviour. The ease of use was the most appreciated characteristic of our system. The set of required parameters was considered sufficiently intuitive and able to provide full control over the generated surfaces. It was pointed out that choosing a suitable similarity measure can be problematic if no explanation is provided. However, after trying out the tool for a short time, the differences between the various measures becomes clear. He also positively commented on the performance of our current prototype. The separate view with the λ_{min} plot was deemed highly valuable for the selection of

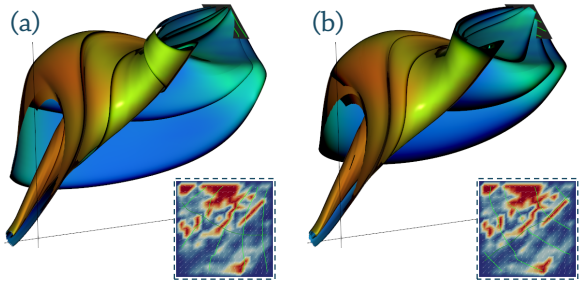


Figure 8: *Same as Figure 4, but the eigenvectors are rotated by 30° (a) and 90° (b) before computing the tensor lines.*

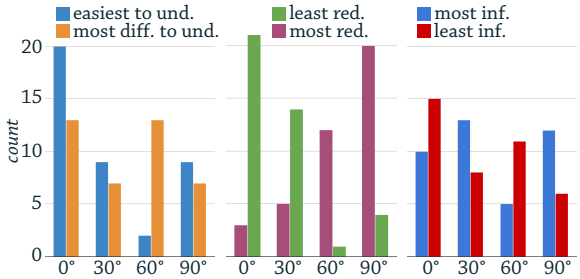


Figure 9: *The results of our informal survey.*

the seeding curve. As a matter of fact, one of his suggestion was to make the separate view more interactive, e.g., by letting the user manipulate the seeding curves on the λ_{min} plot. Another research direction he deemed promising was to adapt the technique for generating a global overview of the dataset. This would be particularly useful when examining an ensemble of simulations over the same spatial domain.

7. Implementation and Performance

Our system is written mainly in C++ and relies on different libraries, such as Qt and VTK. We consider the main contribution of this paper to be the seeding technique rather than its implementation, so we invested a limited amount of efforts in optimizing our software. Here, we report only some indicative timings obtained with our current prototype. The distance matrix, the MDS and the similarity tensor field represent the core of our technique. They are computed partially on the CPU and partially in parallel on the GPU. Our workstation is equipped with a 2.8GHz processor and a GeForce GTX580 graphics card. The distance matrix is relatively inexpensive to evaluate and can be usually computed on the CPU at interactive rates. For instance, even with more that 900 lines (as in Fig. 5), the matrix is computed in less than half a second. The only exception is the Hausdorff distance, which is notoriously expensive. In this case, the distance matrix is partially computed on the GPU by a CUDA kernel. The computation time can still go up to 2 or 3 seconds. For computing the MDS, we utilize the CFMDS algorithm by Park et al. [PSH12]. Their implementation is open source and is based on *CULA*, a GPU-accelerated library for linear algebra. Computing the MDS can take from a few tenths of seconds up to 1 second. For computing the similarity tensors and their spectral decomposition, we implemented a dedicated CUDA kernel. The seeding curves are instead computed on the CPU. These two steps consumes a negligible amount of time (less than 0.01sec). In our experience, the whole process can take from a few tenths of a second up to a few seconds, which still allows for interactive exploration.

8. Summary and Future Work

We have presented a novel technique for automatically seeding multiple stream-surfaces around a point of interest. Our algorithm aims at generating a set of stream-surfaces such that (a) each surface captures a different aspect of the flow behavior, and (b) each surface captures only one aspect of the flow behavior. This is achieved by evaluating the pairwise similarity within a dense set of streamlines. In order to process the resulting similarity matrix, we exploit MDS and tensor analysis in an original way. A preliminary survey indicates that our results are generally easy to understand and free of redundancy. An evaluation by a domain expert confirms the usefulness of our system.

Concerning future developments, we plan to follow the expert's suggestion and enable the possibility of interacting with the λ_{min} plot. It may be also interesting to investigate alternative seeding structures. For instance, a small patch from an As-Perpendicular-As-Possible surface [SRGT12] can be a suitable candidate. Adapting our technique for producing an overview of the dataset can be also accomplished. One option can be clustering the spatial domain and use each cluster as a seeding structure, in analogy with the technique by Edmunds et al. [ELM*12]. Alternatively, we can follow the strategy by Rössl and Theisel [RT12] and seed streamlines throughout the whole domain. The selection of seeding curves can then be guided by the similarity tensor field. A combination of these two ideas could result in a multiscale approach. Adapting our technique to path-lines and path-surfaces is straightforward. For streak-surfaces, it may be interesting to consider an extension of the seeding structure in the temporal dimension. Finally, we plan to further investigate the combination of MDS and tensor analysis. We believe this strategy can lead to interesting results also in different application scenarios.

References

[ABC*96] AURELL E., BOFFETTA G., CRISANTI A., PALADIN G., VULPIANI A.: Growth of noninfinitesimal perturbations in turbulence. *Phys. Rev. Lett. 77* (Aug. 1996), 1262–1265. 5

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

[BVH12] BRAMBILLA A., VIOLA I., HAUSER H.: A hierarchical splitting scheme to reveal insight into highly self-occluded integral surfaces. *Journal of WSCG 20*, 1 (July 2012), 57–64. 3

[BWF*10] BORN S., WIEBEL A., FRIEDRICH J., SCHEUERMANN G., BARTZ D.: Illustrative stream surfaces. *IEEE Trans. on Visualization and Computer Graphics 16* (2010), 1329–1338. 3

[CC10] COX T. F., COX M. A. A.: *Multidimensional scaling*. CRC Press, 2010. 2

[CCK07] CHEN Y., COHEN J. D., KROLIK J. H.: Similarity-guided streamline placement with error evaluation. *IEEE Trans. on Visualization and Computer Graphics 13*, 6 (2007), 1448–1455. 2

[CFM*13] CARNECKY R., FUCHS R., MEHL S., JANG Y., PEIKERT R.: Smart transparency for illustrative visualization of complex flow surfaces. *IEEE Trans. on Visualization and Computer Graphics 19*, 5 (2013), 838–851. 3

[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* (Sept. 2011). 2

[DFH*86] DOMBRE T., FRISCH U., HENON M., GREENE J. M., SOWARD A. M.: Chaotic streamlines in the abc flows. *Journal of Fluid Mechanics 167* (June 1986), 353–391. 7

[DH93] DELMARCELLE T., HESSELINK L.: Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Applications 13*, 4 (1993), 25–33. 6

[ELC*12a] EDMUNDS M., LARAMEE R. S., CHEN G., MAX N., ZHANG E., WARE C.: Surface-based flow visualization. *Computers & Graphics 36*, 8 (2012), 974–990. 2

[ELC*12b] EDMUNDS M., LARAMEE R. S., CHEN G., ZHANG E., MAX N.: Advanced, automatic stream surface seeding and filtering. In *EGUK Theory and Practice of Computer Graphics* (2012), pp. 53–60. 3

[ELM*12] EDMUNDS M., LARAMEE R. S., MALKI R., MASTERS I., CROFT T. N., CHEN G., ZHANG E.: Automatic stream surface seeding: a feature centered approach. In *Computer Graphics Forum* (2012), vol. 31, pp. 1095–1104. 3, 4, 9

[EML*11] EDMUNDS M., MCLOUGHLIN T., LARAMEE R., CHEN G., ZHANG E., MAX N.: Automatic stream surface seeding. In *EuroGraphics 2011 - Short Papers* (2011), pp. 53–56. 3

[ESRT13] ESTURO J. M., SCHULZE M., RÖSSL C., THEISEL H.: Global selection of stream surfaces. *Computer Graphics Forum 32*, 2pt1 (May 2013), 113–122. 3, 4

[GKT*08] GARTH C., KRISHNAN H., TRICOCHE X., BOBACH T., JOY K. I.: Generation of accurate integral surfaces in time-dependent vector fields. *IEEE Trans. on Visualization and Computer Graphics 14*, 6 (2008), 1404–1411. 3

[Hal01] HALLER G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D: Nonlinear Phenomena 149*, 4 (2001), 248–277. 4, 5

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K. I.: Iris: Illustrative rendering for integral surfaces. *IEEE Trans. on Visualization and Computer Graphics 16* (2010), 1319–1328. 3

[HH91] HELMAN J. L., HESSELINK L.: Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications 11*, 3 (1991), 3–46. 2

[HSNHH10] HOTZ I., SREEVALSAN-NAIR J., HAGEN H., HAMANN B.: Tensor field reconstruction based on eigenvector and eigenvalue interpolation. *Scientific Visualization: Advanced Concepts 1* (2010), 110–123. 6

[Hul92] HULTQUIST J. P. M.: Constructing stream surfaces in steady 3d vector fields. In *Proc. IEEE Visualization '92* (1992), pp. 171–178. 3

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. *Visualization in Scientific Computing 97* (1997), 43–56. 2

[KASH13] KRATZ A., AUER C., STOMMEL M., HOTZ I.: Visualization and analysis of second-order tensors: Moving beyond the symmetric positive-definite case. *Computer Graphics Forum 32*, 1 (2013), 49–74. 6

[Kru64] KRUSKAL J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika 29*, 1 (1964), 1–27. 2

[LHS08] LI L., HSIEH H.-H., SHEN H.-W.: Illustrative streamline placement and visualization. In *Proc. IEEE Pacific Visualization Symposium (PacificVis)* (2008), pp. 79–86. 2

[LMGP97] LÖFFELMANN H., MROZ L., GRÖLLER M. E., PURGATHOFER W.: Stream arrows: Enhancing the use of stream-surfaces for the visualization of dynamical systems. *The Visual Computer 13* (1997), 359–369. 3

[LS07] LI L., SHEN H.-W.: Image-based streamline generation and rendering. *IEEE Trans. on Visualization and Computer Graphics 13*, 3 (May-June 2007), 630–640. 2

[MJL*13] MCLOUGHLIN T., JONES M. W., LARAMEE R. S., MALKI R., MASTERS I., HANSEN C. D.: Similarity measures for enhancing interactive streamline seeding. *IEEE Trans. on Visualization and Computer Graphics 19*, 8 (2013), 1342–1353. 2, 5

[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. 2

[MVvW05] MOBERTS B., VILANOVA A., VAN WIJK J. J.: Evaluation of fiber clustering methods for diffusion tensor imaging. pp. 65–72. 2

[PPF*11] POBITZER A., PEIKERT R., FUCHS R., SCHINDLER B., KUHN A., THEISEL H., MATKOVIĆ K., HAUSER H.: The state of the art in topology-based visualization of unsteady flow. *Computer Graphics Forum* (2011), 1789–1811. 4

[PSH12] PARK S., SHIN S.-Y., HWANG K.-B.: CFMDS: CUDA-based fast multidimensional scaling for genome-scale data. *BMC bioinformatics 13*, Suppl 17 (2012), S23. 9

[RT12] RÖSSL C., THEISEL H.: Streamline embedding for 3d vector field exploration. *IEEE Trans. on Visualization and Computer Graphics 18*, 3 (2012), 407–420. 2, 4, 5, 9

[SGRT12] SCHULZE M., GERMER T., RÖSSL C., THEISEL H.: Stream surface parametrization by flow-orthogonal front lines. *Computer Graphics Forum 31*, 5 (Aug. 2012), 1725–1734. 3

[SLM05] SHADDEN S., LEKIEN F., MARSDEN J.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena 212*, 3-4 (2005), 271–304. 5

[SRGT12] SCHULZE M., RÖSSL C., GERMER T., THEISEL H.: As-perpendicular-as-possible surfaces for flow visualization. In *Proc. IEEE Pacific Visualization Symposium (PacificVis)* (2012), pp. 153–160. 9

[Sta99] STALLING D.: *Fast Texture Based Algorithms for Vector Field Visualization*. PhD thesis, University of Berlin, 1999. 3, 6

[TVW99] TELEA A., VAN WIJK J. J.: Simplified representation of vector fields. In *Proc. IEEE Visualization '99* (1999), pp. 35–507. 2

[vW03] VAN WIJK J. J.: Image based flow visualization for curved surfaces. In *Proc. IEEE Visualization 2003* (2003), pp. 123–130. 3

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