Vision, Modeling, and Visualization (2020)
J. Krüger, M. Niessner, and J. Stückler (Eds.)

Static Visualization of Unsteady Flows by Flow Steadification

S. Wolligandt, T. Wilde, C. Rössl, H. Theisel

Abstract

Finding static visual representations of time-varying phenomena is a standard problem in visualization. We are interested in unsteady flow data, i.e., we want to find a static visualization — one single still image — that shows as much of the global behavior of particle trajectories (path lines) as possible. We propose a new approach, which we call steadification*: given a time-dependent flow field* $\mathbf{v}$*, we construct a new steady vector field* $\mathbf{w}$ *such that the stream lines of* $\mathbf{w}$ *correspond to the path lines of* $\mathbf{v}$*. With this, the temporal behavior of* $\mathbf{v}$ *can be visualized by using standard methods for steady vector field visualization. We present a formal description as a constraint optimization that can be mapped to finding a set cover, a NP-hard problem that is solved approximately and fairly efficiently by a greedy algorithm. As an application, we introduce the first 2D image-based flow visualization technique that shows the behavior of path lines in a static visualization, even if the path lines have a significantly different behavior than stream lines.*

1. Introduction

Many objects considered in Visualization are time-dependent, i.e., they change appearance and behavior over time. Creating visualizations of time-dependent phenomena has always been a main challenge in visualization. An obvious and straightforward approach is the use of animations: the object is visualized at a certain time t, where t varies monotonically or is varied interactively. While this gives a natural interpretation of the time-parameter — time is mapped to time — animations come with an overhead in computation, storage, and perception. For this reason, the search for static visualizations of time-dependent objects has always been a hot topic in visualization [AMST11]. Examples include static visualizations of time-dependent trees [KW19], stories [LWW*13], graphs [BBDW17], and scalar fields [BGH01].

In this paper, we focus on static visualizations of time-dependent vector fields, which usually describe flows. Given an unsteady vector field $\mathbf{v}(\mathbf{x},t)$, we search for static visualizations that describe as much as possible of the dynamic behavior of $\mathbf{v}$. In order to refine this rather general description, we introduce the following concepts.

Time-local vs time-global visualizations: A static visualization is *time-local* if it encodes only information of $\mathbf{v}$ at a certain time t_0. Time-local visualizations cannot encode the dynamic behavior of an object (unless t_0 is varied in an animation, which is not considered here). Contrarily, a *time-global* visualization encodes information of $\mathbf{v}$ from the complete temporal domain. As such, time-global techniques have the potential to encode dynamic behavior of $\mathbf{v}$. Note that there are also intermediate cases where in addition to considering $\mathbf{v}$ at t_0, either time-derivatives of $\mathbf{v}$ or information from a time interval $[t_0, t_0+\Delta t]$ are considered. We call these techniques *semi-time-local*.

Characteristic integral curves in $\mathbf{v}$: A common class of flow visualization techniques shows integral curves, namely stream lines, path lines, streak lines, and time lines. While stream lines are easy to construct, they fail to show the dynamic behavior of $\mathbf{v}$ because they are time-local. Moreover, stream lines do not exhibit a physical meaningful property of the flow. Contrarily, path lines describe the trajectories of massless particles, which makes them a perfect candidate for visualizing the dynamic behavior of $\mathbf{v}$. Note that streak lines and time lines are also time-global, but the cardinality of streak lines and path lines is much higher than for path lines: Through every point $(\mathbf{x},t)$ passes only one single path line but a whole family of streak lines or time lines.

Based on these concepts we refine our problem: We search for static time-global visualizations that show the behavior of particle trajectories — i.e., path lines — of $\mathbf{v}$. The core idea of our approach can be phrased in one sentence: Given an unsteady vector field $\mathbf{v}(\mathbf{x},t)$, find a steady vector field $\mathbf{w}(\mathbf{x})$ such that the stream lines of $\mathbf{w}$ are the path lines of $\mathbf{v}$. The motivation for this approach comes from the fact that Flow Visualization is much further developed in the investigation of stream lines than of path lines. There exists a number of well-established algorithms that work exclusively on stream lines, and our approach makes them available to the analysis of path lines. This way path line visualization may become easier accessible. In particular, this approach provides solutions to previously unsolved problems such as 2D image-based flow visualization that depicts the behavior of particles in one single still image.

We show that finding such vector field $\mathbf{w}$ is a nontrivial problem since it requires a multi-objective optimization in a huge search space. The optimization problem can be expressed as set cover problem, and we propose a greedy algorithm for finding an approximate solution. For the actual visualization of $\mathbf{w}$ we adapt standard methods, in particular the well-known LIC technique with an additional special treatment of discontinuities of $\mathbf{w}$ as well as a color coding of the time. To the best of our knowledge, this is the first

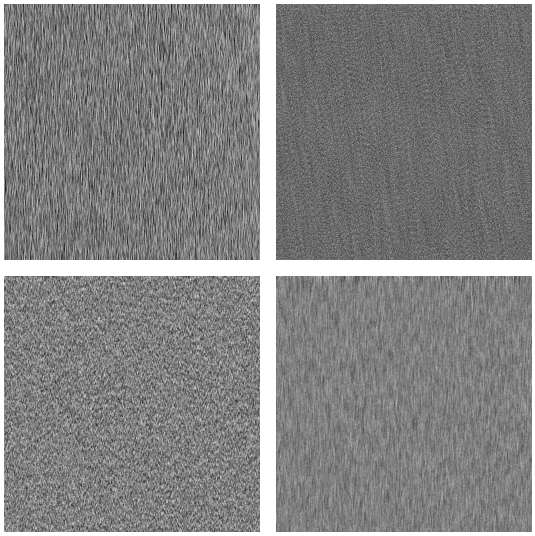

Figure 1: *Static visualizations of the flow* $\mathbf{v}_1$ *defined in* (1) *for* $t = 0$*. Top left:* UFLIC *[SK97] (*$\tau = 0.06$*), Top right:* UFAC *[WEE03] (*$\tau = 0.2$*), and Bottom:* IBFV *[Wij02] (*$\tau = \pi$*) for step width* 0.01 *(left) and* 0.05 *(right).*

image-based technique that shows the behavior of particle paths in a single image.

2. Related work

This section gives a brief review of techniques for the visualization of unsteady flows. Such techniques can be classified into image or texture based methods and methods that show path lines.

Texture based unsteady flow visualization Texture based techniques have a long history in flow visualization. Originally developed for 2D steady flows, a variety of extensions to unsteady flows, flows on surfaces, or 3D flows have been proposed. A comprehensive state-of-the-art report is given by Laramee et al. [LHD*04]. Here we restrict ourselves to techniques specialized on *2D unsteady flows*.

Texture based techniques can be classified into spot noise, line integral convolution (LIC), and texture advection [LWSH04]. De Leeuw and van Liere [LL99] present an unsteady spot noise approach where spots are modeled following the particle paths. The first approach to extend LIC to unsteady flows is the unsteady LIC approach by Forssell et al. [For94, FC95]. There, the convolution kernel follows path lines instead of stream lines. The temporal coherence of unsteady LIC was improved by the UFLIC approach by Shen and Kao [SK97], which scatters particles along path lines and spreads this way their contributions. Later, more efficient versions have been developed like AUFLIC [LMI02] or GPUFLIC [LTH06]. DLIC [Sun03] was developed by Sundquist as a framework for the advection of stream lines over particle paths.

Texture based techniques are based on a forward or backward advection of noise textures. Image-based flow visualization by van Wijk [Wij02] is a rather general approach, which is based on forward advection and blending of subsequent images. This approach is able to resemble several other techniques by choice of parameters. Lagrange-Eulerian Advection (LEA) by Jobard et al. [JEH00] combines Eulerian and Lagrangian view points in a texture advection approach. UFAC by Weiskopf et al. [WEE03] advects and deforms stream line patterns steered by the trajectories of path lines. Carnecky et al. [CSFP12] introduce a multilayer dense flow visualization for unsteady flows on time-dependent surfaces.

Path line seeding Another general approach to static visualization of time-dependent flows is the selection and rendering of a carefully chosen set of path lines. While stream line selection [JL97] is a well-established approach that has been extended to time-coherent stream lines in unsteady flows (see [JL00]), the selection and rendering of suitable path lines is less intensively researched. McLoughlin et al. [MET*15] present an approach to path line seeding based on local importance measures. Weinkauf et al. [WTS12] select path lines with a minimal number of intersections in the visualization. Marchesin et al. [MCHM10] present view dependent stream line selection by adding 3D stream lines from a precomputed set. Their method considers both, the footprint of the already chose lines and the local properties of the new lines. Günther et al. [GRT14] aim at rendering large sets of path lines and resolve visual clutter by computing opacity values based on a global view-dependent optimization. Hlawatsch et al. [HSJW14] present glyph representations of selected path lines.

3. An analysis of image-based visualization of 2D unsteady flows

Image-based techniques are an extremely successful tool for the visualization of *steady* 2D vector fields, and consequently they have been extended and emerged as a standard method for visualizing *unsteady* 2D flows. Given this practice, the following statement may come surprisingly:

Statement 1 Existing techniques for texture based flow visualization are unable to show the behavior of path lines in a static visualization unless they are similar to stream lines.

This means that existing texture based techniques cannot represent the dynamic behavior of an unsteady flow in a static visualization. At first, the statement seems to be counterintuitive and in conflict with common practice as we see many impressive texture based unsteady flow visualizations in the literature. We argue that this impression is due to the fact the considered flows often show "low unsteadiness", i.e., stream lines and path lines are rather similar. A simple examples shows that existing images based techniques must fail, whenever this is not the case.

Consider the unsteady field $\mathbf{v}_1 : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ with

$$\mathbf{v}_1(\mathbf{x},t) = \begin{pmatrix} \cos t \\ \sin t \end{pmatrix} . \tag{1}$$

The following properties hold for $\mathbf{v}_1$:

- All stream lines of $\mathbf{v}_1$ are straight lines.
- All path lines of $\mathbf{v}_1$ are unit circles.
- $\mathbf{v}_1$ is a flow, i.e., it satisfies the incompressible Navier-Stokes equation with the pressure field $p(\mathbf{x},t) = x\sin t - y\cos t$ with $\mathbf{x} = (x,y)^T$.

This makes $\mathbf{v}_1$ a perfect benchmark for any texture based technique: If the visualization shows unit circles or arcs of unit circles, we see

path lines. If it shows straight lines only, the technique shows exclusively stream lines and is therefore unable to show the dynamic behavior of the flow.

Figure 1 shows *static* visualizations of $\mathbf{v}_1$ generated by UFLIC [SK97], LEA [WEE03], and IBFV [Wij02]: they clearly fail to present properties of the flow. However, they are successful for *dynamic* visualizations as shown by the animation in the accompanying **video**.

We close this sections with two remarks. Firstly, several informal statements that point into the same direction as statement 1 were made in the literature. For instance, Shen and Kao [SK97] considered also highly unsteady flows and observed that the more rapidly the direction of the flow changes the more blurring occurs in the visualization. In contrast to such empirical observations, the benchmark flow $\mathbf{v}_1$ allows for a formal study and formulation of statement 1. Secondly, statement 1 holds only for static visualizations — a single still image — not for sequences of images (e.g.displayed as time-varying animations). In fact, information about the behavior of path lines can be obtained exclusively from a temporal coherence without any spatial coherence. An example would be observing the movement of some points over time. In this case, a still image would show a set of individual points. So, *existing techniques use temporal coherence*, and this way they have the potential of showing paths of particles.

4. Formal problem statement and analysis

We consider a flow

$$\mathbf{v} : \mathcal{D} \times [t_{\min}, t_{\max}] \to \mathcal{D} \quad \text{with} \quad \mathcal{D} \subseteq \mathbb{R}^2 ,$$

i.e., the spatial domain is possibly unbounded whereas the temporal domain is restricted to the interval $[t_{\min}, t_{\max}]$. The *path line* of $\mathbf{v}$ starting at $(\mathbf{x}, t)$ is the parametric curve $\mathbf{r}(\mathbf{x}, t, \tau)$ that is the solution to the initial value problem

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{r}(\mathbf{x}, t, \tau) = \mathbf{v}(\mathbf{r}(\mathbf{x}, t, t+\tau)) \quad \text{with} \quad \mathbf{r}(\mathbf{x}, t, 0) = \mathbf{x}$$

and $\tau \in [t_{\min} - t, t_{\max} - t]$. The differential equation describes the motion of a massless particle in $\mathbf{v}$ starting at $(\mathbf{x}, t)$.

The *stream line* of a steady vector field $\mathbf{w} : \mathbb{R}^2 \to \mathbb{R}^2$ starting at $\mathbf{x}$ is the parametric curve $\mathbf{s}(\mathbf{x}, \tau)$ that solves

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{s}(\mathbf{x}, \tau) = \mathbf{w}(\mathbf{s}(\mathbf{x}, \tau)) \quad \text{with} \quad \mathbf{s}(\mathbf{x}, 0) = \mathbf{x} ,$$

for $\tau \in \mathbb{R}$. Based on these definitions, we formulate steadification as the following problem:
Given the unsteady flow $\mathbf{v}(\mathbf{x}, t)$, find fields $\mathbf{w}(\mathbf{x})$ and $s(\mathbf{x}), \tau_{\min}(\mathbf{x})$, and $\tau_{\max}(\mathbf{x})$ such that for every $\mathbf{x} \in \mathcal{D}$

$$\mathbf{s}(\mathbf{x}, \tau) = \mathbf{r}(\mathbf{x}, s(\mathbf{x}), \tau) \quad \text{for} \quad \tau_{\min}(\mathbf{x}) \le \tau \le \tau_{\max}(\mathbf{x}) . \tag{2}$$

In this problem statement, the unknown $\mathbf{w}$ is the steady vector field whose stream lines correspond to path lines of $\mathbf{v}$. The scalar field s describes a start — or seed — time of the path line in $\mathbf{v}$ that is to be matched by a stream line in $\mathbf{w}$. Finally, the time interval $[\tau_{\min}, \tau_{\max}]$ restricts the correspondence of path lines and stream lines in time. Together, the vector field $\mathbf{w}$ and the scalar fields $s, \tau_{\min}$, and $\tau_{\max}$ represent a *steadification* of $\mathbf{v}$. We will treat them as unknowns or degrees of freedom in a search for an optimal steadification.

In the following we analyze properties of steadification in order to characterize an optimization problem. We start with the remark that a steadification is *not unique*. This is easy to see: In (2), a path line $\mathbf{r}$ is seeded at $(\mathbf{x}, s(\mathbf{x}))$. The same path line can be represented using a different seed, where any point on the curve $(\hat{\mathbf{x}}, \hat{s}(\hat{\mathbf{x}})) \in \mathbf{r}$ is a possible seed. Furthermore, the following identities must hold for all $\mathbf{x} \in \mathcal{D}$ and $\tau \in [\tau_{\min}(\mathbf{x}), \tau_{\max}(\mathbf{x})]$:

$$\mathbf{v}(\mathbf{r}(\mathbf{x}, s(\mathbf{x}), \tau), s(\mathbf{x}) + \tau) = \mathbf{w}(\mathbf{r}(\mathbf{x}, s(\mathbf{x}), \tau)) \tag{3}$$

$$\tau_{\min}(\mathbf{r}(\mathbf{x}, s(\mathbf{x}), \tau)) = \tau_{\min}(\mathbf{x}) - \tau \tag{4}$$

$$\tau_{\max}(\mathbf{r}(\mathbf{x}, s(\mathbf{x}), \tau)) = \tau_{\max}(\mathbf{x}) - \tau . \tag{5}$$

The first condition (3) implies that the path line $\mathbf{r}(\mathbf{x}, s(\mathbf{x}), \tau)$ does not have self-intersections for $\tau \in [\tau_{\min}(\mathbf{x}), \tau_{\max}(\mathbf{x})]$. Otherwise, $\mathbf{w}$ would be overdetermined. Since, in general, path lines *can* have self-intersections, solutions to (2) are generally not continuous, i.e., the respective fields show *cuts* that separate continuous regions in space-time. The choice of cuts leaves another degree of freedom that makes the problem underdetermined. In summary, the solution of (2) is not unique.

Therefore, we define additional constraints that characterize favorable solutions from the infinite set of feasible solutions:

1. Prefer *long path lines*: $\tau_{\max}(\mathbf{x}) - \tau_{\min}(\mathbf{x})$ should be maximized.
2. Prefer *curved path lines*: the generated steady flow $\mathbf{w}(\mathbf{x})$ should not be laminar
3. Prefer *large continuous regions*: The number and length of cuts should be minimized.

The first requirement is motivated by the fact that for short integration times stream lines and path lines tend to look similar, which would focus on local "snapshots" of stream line behavior at certain times. The longer the integration time the more of the temporal dynamics are captured. The second requirement ensures that more interesting regions of the flow are covered. The last requirement is motivated by the fact that $\mathbf{v}$ is assumed to be continuous, hence the derived field $\mathbf{w}$ should also be continuous or at least as continuous as possible. In the visualization, larger space-time continuous regions give a more global insight into the flow and are easier to interpret than a solution that is "mosaic-like" or has cuts that are rather long compared to the enclosed area.

Obviously, the three requirements are competing: the longer the integration time and the higher the curvature for path lines, the higher the probability of a self-intersection and hence the need for introducing discontinuities. We need to find a reasonable balance. We also note that the additional requirements provide no guarantee that there exists a unique optimal solution. However, given that the search space is huge, an exhaustive search is unrealistic anyway. In the following, we present a discretization of the search space and an algorithm that uses a greedy strategy. It turns out that, in practice, we find local optima that lead to good flow steadification that give meaningful static visualizations.

5. Flow steadification as an optimization problem

Flow steadification can be modeled as a combinatorial optimization problem: the *Set Cover Problem*, one of Karp's 21 NP-complete problems [Kar72].

The *Set Cover Problem* can be summarized as follows: Given a set $\mathcal{U}$ and n subsets $\mathcal{S}_i \subset \mathcal{U}$ with $i = 1,\dots,n$, decide if there exists a set cover $\mathcal{C} = \{\mathcal{S}_j : j \in \{1,\dots,n\}\}$ of size $|\mathcal{C}| = k$ such that $\bigcup_{\mathcal{S}_j \in \mathcal{C}} \mathcal{S}_j = \mathcal{U}$.

We solve this problem with a greedy algorithm [Chv79]: The algorithm starts with the set of all subsets $\mathcal{W} = \{\mathcal{S}_i : i = 1\dots,n\}$ and an initial state $\mathcal{C} = \emptyset$. Then, subsets $\mathcal{S}_i \in \mathcal{W}$ are iteratively selected, removed from $\mathcal{W}$ and added to $\mathcal{C}$ until either $\mathcal{W} = \emptyset$ or $\bigcup_{\mathcal{S}_j \in \mathcal{C}} \mathcal{S}_j = \mathcal{U}$, i.e., a set cover is found. The policy for selection is picking the subset $\mathcal{S}_i \in \mathcal{W}$ that covers the most uncovered elements, i.e., that minimizes $|\mathcal{U} \setminus (\mathcal{C} \cup \mathcal{S}_i)|$ or equivalently maximizes $|\mathcal{C} \cup \mathcal{S}_i|$.

An extension of the algorithm equips each $\mathcal{S}_i \subset \mathcal{U}$ with a weight w_i. The selection policy changes to picking the subset $\mathcal{S}_i \in \mathcal{W}$ that maximizes the accumulated weight $\sum_{\mathcal{S}_j \in \mathcal{C} \cup \mathcal{S}_i} |\mathcal{S}_j| \, w_j$.

We utilize this algorithm as follows. The domain $\mathcal{D}$ is discretized into cells in a regular grid D of dimensions $D_w \times D_h$, i.e., $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n\}$ consists of all cells represented by their center points $\mathbf{x}_i$. The initial set $\mathcal{W}$ is constructed from another regular grid G that is superimposed over D. This grid is coarser with a user defined resolution $G_w \times G_h$ with $G_w \leq D_w$ and $G_h \leq D_h$. We consider the edges that span cells and their diagonals in G. Each edge $e_k \in G$ is used as a seed curve for integration of a path surface $\mathcal{R}_k$ in the flow $\mathbf{v}(\mathbf{x},t)$. The start time t_0 is fixed to the mean $\frac{1}{2}(t_{\min} + t_{\max})$, and integration times are $\tau = \frac{1}{2}(t_{\min} + t_{\max})$ for forward and $-\tau$ for backward integration. We remark that a similar discretization and surface integration from seed edges was used for 3d stream surface selection in [MSRT13]. The projection of a path surface $\mathcal{R}_k$ to the spatial domain covers a set of discrete cells in D. This simple projection, however, disregards the fact that the spatial pixel can be covered multiply at different times, because path lines can intersect in space. In other words, there may appear multiple different vectors $\mathbf{v}(\mathbf{x}_i,t)$ at each spatial position $\mathbf{x}_i \in D$ at different times t. Let n_i denote the number of vectors sampled at $\mathbf{x}_i$. We distinguish three cases when probing $\mathcal{R}_k$ at $\mathbf{x}_i$: First, $\mathbf{x}_i$ is not covered before iteration j by any path surface, which means $n_i = 0$. If the path surface $\mathcal{R}_j$ covers $\mathbf{x}_i$ then t_{top} will be the time when $\mathcal{R}_j$ passes $\mathbf{x}_i$ with its *greatest* integration length $|\tau|$ at $\mathbf{x}_i$. Second, $n_i = 1$, and a path surface $\mathcal{R}_k$, $k \geq j$ passes $\mathbf{x}_i$ again then t_{back} will be the time when $\mathcal{R}_k$ passes $\mathbf{x}_i$ with its *smallest* integration length $|\tau|$. Third $n_i \geq 2$, and $\mathbf{x}_i$ is passed by a path surface $\mathcal{R}_m$, $m \geq k$, then the count n_i is just incremented by one. These three cases can easily be implemented by rendering path surfaces $\mathcal{R}_k$ and testing cells $\mathbf{x}_i$.

Finally, we define the weight w_i at $\mathbf{x}_i$ as follows:

$$w_i = \begin{cases} a_i & \text{for } n_i = 1 \\ b_i & \text{for } n_i = 2 \\ 0 & \text{else} \end{cases}$$

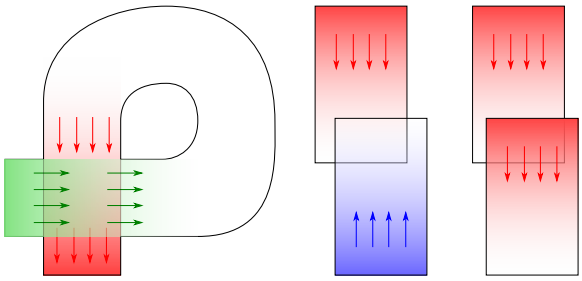

Figure 2: *Illustration of weights w_i in (6) for the case that x_i is covered twice. Left: $sin^2\varphi = 1$. Swirling behavior can be well perceived if a region is coverered exactly twice. Center and right: $sin^2\varphi = 0$.*

where

$$\begin{aligned} a_i &= \tau_i \cdot \min\{1, \kappa_i/\kappa_{\text{ref}}\} \,, \\ b_i &= a_i \cdot \sin^2 \varphi_i \,, \qquad \text{and} \\ \varphi_i &= \angle(\mathbf{v}(\mathbf{x}_i, t_{\text{top}}), \mathbf{v}(\mathbf{x}_i, t_{\text{back}})) \,. \end{aligned} \tag{6}$$

Here, τ_i denotes the integration time until $\mathbf{x}_i$ is reached, and κ_i is the unsigned curvature of the path line at $\mathbf{x}_i$. Both refer to t_{top} if $n_i = 2$. The reference curvature κ_{ref} is a user-defined parameter.

The rationale of this choice is as follows: Our first goal is to cover the whole domain. However, a greedy selection for just the maximum additional cover does not take into account properties of the flow and would result in an arbitrary choice and thus probably visualize "uninteresting" path lines. Instead, the weighted set cover prefers longer integration times and high curvature and penalizes (anti-)parallel directions when $\mathbf{x}_i$ is covered exactly twice. If $\mathbf{x}_i$ is covered more than twice, "clutter" is assumed. The local curvature is scaled and clamped, to avoid extreme weights and artifacts by local curvature outliers. The responsible user parameter κ_{ref} also balances the influence of curvature versus integration time. If $\mathbf{x}_i$ is covered exactly twice, the angle θ_i between $\mathbf{v}(\mathbf{x}_i, t_{i,\text{top}})$ and $\mathbf{v}(\mathbf{x}_i, t_{i,\text{back}})$ is used to map the weight between 0 and 1 for parallel and perpendicular flow directions, respectively. The angle is taken into account for the weight to ensure that swirling behavior is captured, as this can be well perceived when exactly two areas are overlapping (see figure 2).

This weighting scheme does not yet take into account spatial coherence, i.e., the algorithm may select many fragments of path surfaces, which results more cuts and thus in a cluttered view. To avoid this, we "boost" the total weight $W_i = \sum_{\mathcal{R}_j \in \mathcal{C} \cup \mathcal{R}_i} w_j$ of a path surface $\mathcal{R}_i$: We multiply W_i by a constant $c > 1$ if the seed curve of $\mathcal{R}_i$ can be *joined* smoothly with that of another path surface $\mathcal{R}_j \in \mathcal{C}$. This results in testing for neighboring seed edges, where we assume a sufficiently smooth joint if the smaller angle θ between two edges is $\theta \geq 135°$. Otherwise, if $\theta \leq 45°$, or if a seed edge touches or intersects an already *joined* seed curve we set $W_i = -\infty$. Without this penalty, path surfaces with low weight W_i that would introduce clutter would be "boosted" and preferred for set cover. Furthermore, we exclude diagonal seed edges that cross other diagonal seed edges also with a penalty $W_i = -\infty$ (figure 3 bottom center). The different cases for "boosting" weights are summarized and illustrated in figure 3. Section 7 studies the effect of varying the user parameters, the boost factor c and the reference curvature κ_{ref}.

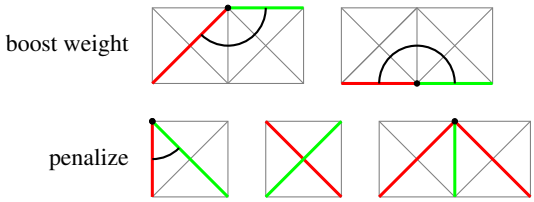

Figure 3: *Configurations of neighboring seed edges in the grid G. Green denotes a newly added seed edge, and red represents an already present seed curve. Top row: "boost" W_i by factor c if a new seed edge "smoothly" continues another seed curve with angle of $\theta = 135°$ or $\theta = 180°$. Bottom row: Set $W_i = -\infty$ if either the angle between a newly added and an already present seed curve is $\theta \leq 45°$ (left), or if a new seed curve crosses another seed edge (center), or if a new seed curve would intersect an already continued seed curve (right).*

The implementation of the overall algorithm is not difficult (see Algorithms 1 and 2). We use an adaptive fourth-order Runge-Kutta scheme for the integration of path surfaces. Note that the partial surfaces that are integrated from single seed edges can be stored and reused for evaluation by the algorithm similarly as proposed in [MSRT13]. The path surfaces are rendered such that for every pixel the weight w_i is computed and accumulated to get W_i (before boost) on the GPU.

The result of this steadification by finding a set cover is a sampling of the steady vector field $\mathbf{w}$ on the domain grid D, i.e., a $D_w \times D_h$ "image" of the selected velocity vectors. This vector field generally shows discontinuities at cuts between different projected path surfaces. We do not need an explicit encoding of location and topology of these cuts.

Algorithm 1: Framework of the greedy set cover algorithm.

```
Algorithm: greedy_set_cover
Input  : discretization D, desired_coverage, c, κ_ref
Output: steadification of w
coverage ← 0; best_weight ← −∞;
best_edge ← NONE; used_edges ← ∅;
unused_edges ← {e ∈ D.edges};
do
    best_edge ← NONE;
    best_weight ← −∞;
    for e ∈ unused_edges do
        w ← measure_weight(κ_ref, e, used_edges);
        // (see Algorithm 2 for description measure_weight)
        w ← w · boost(w, c, e, used_edges);
        // (see Figure 3 for explanation of boost)
        if w > best_weight then
            best_weight ← w; best_edge ← e;
        end
        if best_edge != NONE then
            used_edges.enqueue(best_edge);
            unused_edges.remove(best_edge);
            coverage ← update_coverage(used_edges);
        end
    end
while (coverage < desired_coverage
  AND best_edge != NONE);
// generate w, s, τ_min and τ_min from the path surfaces seeded at the
  edges e ∈ used_edges
```

Algorithm 2: Calculation of a path surface's weight depending on previously added path surfaces seeded at the grid edges stored in "unused_edges". A rasterization R consists of fragments at discrete positions. A fragment f saves a list of attributes describing path surfaces at their corresponding spatial and temporal position. While rasterizing a path surface each fragment's list gets sorted from $|\tau|$ to 0, where the first element in f is related to the highest $|\tau|$.

```
Algorithm: measure_weight
Input  : κ_ref, unused_edge, used_edges
Output: edge_weight
// Initialize rasterization R with empty lists.
// Initialize weight_map with {0,0,...}.
for e ∈ {used_edges, unused_edge} do
    P ← integrate_pathsurface(e);
    PR ← rasterize(P);
    R.insert(PR);
end
for f ∈ R do
    a ← f.first.τ * min(1, f.first.κ / κ_ref);
    phi ← angle(f.first.velocity, f.last.velocity);
    b ← a · sin²(phi);
    if f.size == 1 then
        weight_map[f] ← a;
    else if f.size == 2 then
        weight_map[f] ← b;
    else
        weight_map[f] ← 0;
    end
end
edge_weight ← accumulate(weight_map);
```

6. Static Visualization

We utilize a modified line integral convolution (LIC) [CL93] algorithm for static visualization. We apply LIC on the normalized vectors of the field $\mathbf{w}$ using a GPU-implementation with a fixed-size fourth-order Runge-Kutta integration and a steps size equal to half pixel size. We consider a pixel as a discontinuity when the forward and backward difference in x- and y-direction at the position of this pixel differs by a value of 0.5. In contrast to the standard LIC algorithm, we stop integration between adjacent cells $\mathbf{x}_i$ and $\mathbf{x}_j$ of the stream line if either $\mathbf{x}_i$ or $\mathbf{x}_j$ are marked discontinuous. Effectively, this means that there is no integration and no convolution across cuts, which are treated "image based". In addition, we color-code the integration time τ_i using a predefined color — white in our examples — when x_i is uncovered.

7. Results

We demonstrate steadification for two analytic flows and three data sets from simulations and provide a parameter study for the 'boost" factor c and reference curvature κ_{ref}. All timings were measured on a computer with an Intel Core i7-7700K CPU at 4.2GHz, 32GB memory and a NVIDIA GeForce GTX 1080 GPU.

Test field $\mathbf{v}_1$. The test data set $\mathbf{v}_1$ is introduced and discussed in Section 3. There we also have shown that existing texture based flow visualization techniques are unable to show the behavior of

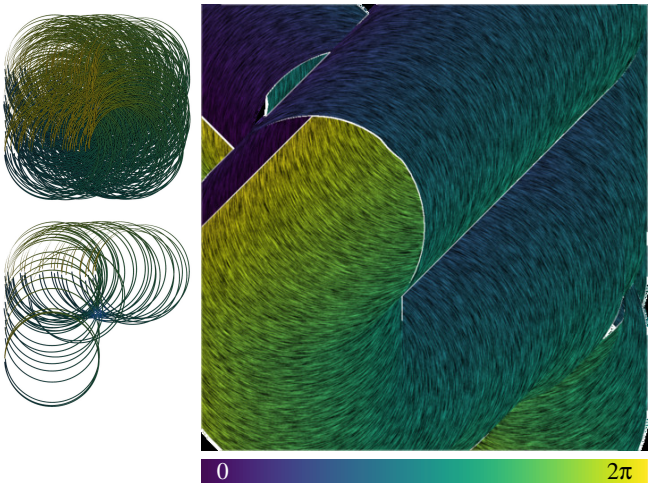

Figure 4: *Test field* $\mathbf{v}_1$ (1)*: Left: View-dependent reduction of random path lines using [MCHM10] (top:* 100%*, bottom:* 10%*). left: Steadification (*21×21 *seed grid,* $\kappa_{ref} = 15$*,* $c = 2$*. iterations), color coded with time* t*.*

path lines in a static image. Also, a naive seeding of path lines fails due to the large number of intersections of path lines. Figure 4 (left column) shows path lines with different densities, the bottom row shows careful selections of lines by [MCHM10].

While in both path line visualizations the multiple intersections of the path lines indicate a highly unsteady behavior, a further characterization of this behavior hardly seems possible.

Contrarily, our visualization (Figure 4, right) clearly shows distinct and coherent arcs of unit circles as path lines. Also, the discontinuities of $\mathbf{w}$ across cuts are visible. The computing time for the steadification was approximately 5 minutes. As — to the best of our knowledge — this is the only existing texture based technique that clearly shows arcs of unit circles and therefore the behavior of path lines. Due to the simplicity of $\mathbf{v}_1$, the cuts — i.e., the linear regions of discontinuities in $\mathbf{w}$ — do not have a physical meaning. They result from the specific parameter setting and initial state of the set cover algorithm.

To the best of our knowledge, $\mathbf{v}_1$ has not been used in the literature as benchmark data set. Due to its simplicity and clear distinction of stream lines and path lines, we consider the introduction of this benchmark data set a contribution of this paper.

Double Gyre. The double gyre is a synthetic data set, which was introduced by Shadden et al. [SLM05]. Particles advected in the domain $[0,2] \times [0,1]$ never leave this domain. Since its introduction, the double gyre has become a successful standard data set for flow analysis, in particular Lagrangian coherent structures (LCS) [Hal15] and Finite Time Lyaponov fields [HY00, Hal01]. For this reason, many versions of LCS visualizations of the double gyre can be found in the literature. Contrarily, the data set is usually visualized only by standard techniques for certain time slices. Static visualizations of the double gyre which exhibit the dynamic behavior of path lines are rarely shown in the literature. Figure 6 (top left) shows a LIC visualization of a (steady) time slice that is unable to depict the dynamic behavior. UFLIC (top right) looks similar to standard LIC. A random selection of path lines (middle left) shows that the data set is indeed time-varying. Middle right shows careful selections of lines by [MCHM10]. It does, however, not provide information about the global dynamic behavior. Our result (Figure 6, bottom) clearly shows long particle trajectories in the texture that is coherent for long integration times. Also, regions of self-intersection path lines are visible, and along the cuts discontinuities of $\mathbf{w}$ are clearly visible. The computing time was approximately 5 minutes.

Cavity flow. The cavity data set is a vector field describing the flow over a 2D cavity. This data set was kindly provided by Caraballo et al. [CSD03] and B. Noack and I. Pelivan. 1 000 time steps were simulated using the compressible Navier-Stokes equations. The flow exhibits a non-zero divergence inside the cavity, while outside the cavity the flow tends to have a quasi divergence-free behavior. The data is almost — but not perfectly — periodic with a period of about 100 time steps in length, and only the first 100 time steps are shown. This gives a resolution of $256 \times 96 \times 100$. In visualization, this data set has been analyzed in [TWHS05, WRT18]. Cavity flows, i.e., laminar flows passing over an open cavity, are of interest in many applications in engineering, ranging from the small cavities due to gaps in the body work of vehicles, over the shapes of river channel beds, or cargo bays on aircraft, to the large scale flows in urban street canyons.

Figure 7 (top) shows a LIC image of a time slice at $t = 5$. Our steadification (bottom) shows a fairly steady flow outside the cavity but a highly unsteady behavior — including many intersection path lines — inside the cavity. This is an example with a non-convex domain. The computing time was approximately 5 minutes.

User parameters. The steadification algorithm requires several parameters. The first choice are the dimensions of the seed grid G. We show a variety of grids for the different examples. Generally, this grid is chosen rather coarse, e.g., in the order of 30 to 100 for the longer side of a rectangular domain. We found that the algorithm is not very sensitive to the particular choice. However, a very coarse seed grid is likely to restrict the search space too much, while an excessively fine grid leads to a significant increase in computation times without significant benefit.

There are essentially two scalar user-parameters, the boost factor $c > 1$ and the reference curvature κ_{ref}. Higher c leads to longer seed curves and thus larger continuous regions at the cost of losing temporal detail. Higher κ_{ref} prefers path lines with higher curvature and thus more possibly interesting detail at the cost of more discontinuities. Figure 5 shows a table with different choices, which illustrates these properties for the DOUBLE GYRE. As a recommendation, the boost factor should be selected at least slightly above 1, we generally recommend to start with 1.5. The reference curvature depends on the absolute dimensions of the domain — a unit circle has curvature 1. The particular choice depends on the data set. We recommend a value of $10 - -15$ for a domain sized in the order of the unit square. Table 1 shows the iteration count of the greedy set cover algorithm required to generate these results. The numbers show that the choice of parameters does not have very large impact, the numbers differ approximately by a factor of two.

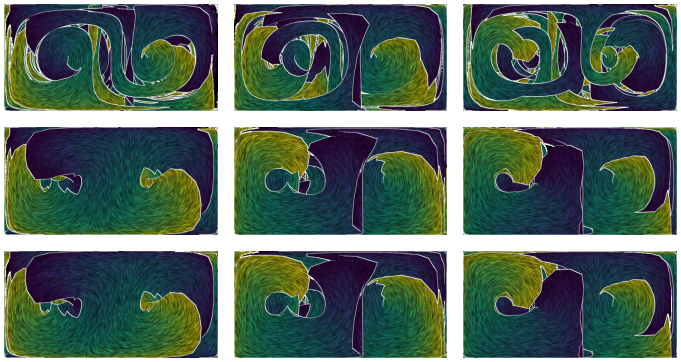

Figure 5: DOUBLE GYRE. *Results for varying boost factor c (top: $c = 1.0$, center: $c = 1.5$, bottom: $c = 2.0$) and reference curvature κ_{ref} (left: $\kappa_{ref} = 5$, right: $\kappa_{ref=10}$, right: $\kappa_{ref} = 15$).*

c	$\kappa_{\text{ref}} = 5$	$\kappa_{\text{ref}} = 10$	$\kappa_{\text{ref}} = 15$
1.0	207	212	211
1.5	208	129	91
2.0	205	127	87

Table 1: *Number of iterations for convergence for generating the results in figure 5.*

8. Discussion and limitations

In this section we discuss various aspects of the steadification approach.

Is it worth the effort? Flow steadification is fairly expensive and requires significant computation time. We argue that steadification is a preprocess that is carried out and stored once for an unsteady data set. Once computed, it can be visualized by — possibly interactive — standard techniques for steady flows.

2D vs. 3D Flow steadification aims at 2D steady visualizations of a 2D unsteady flow. An alternative approach is to interpret the 2D unsteady flow as 3D steady flow by introducing time as third spatial dimension. While this allows for making use of standard 3D steady visualization techniques, it also inherits their potential problems: ambiguities in the projection and visual clutter. The question of 2D techniques vs. 3D techniques has been intensively discussed in the visualization community, leaving a number of arguments for using 2D visualizations whenever possible. For 2D time-dependent flows, this means that flow steadification is a 2D technique supplementing 3D steady techniques.

Can flow steadification miss features? Flow steadification is not a feature based technique. Features are not explicitly extracted and are therefore not guaranteed to be presented. An example where feature omission cannot be avoided is when different features pass the same location at different times. Then it is clear that only at most one of them can be seen in the steadification.

There is, however, some evidence that features are visible in a flow steadification: for instance in regions of vortices, the path surfaces show an extreme curvature, which leads to a larger number of integration stops and cuts — i.e., linear regions of discontinuities — in the steadification.

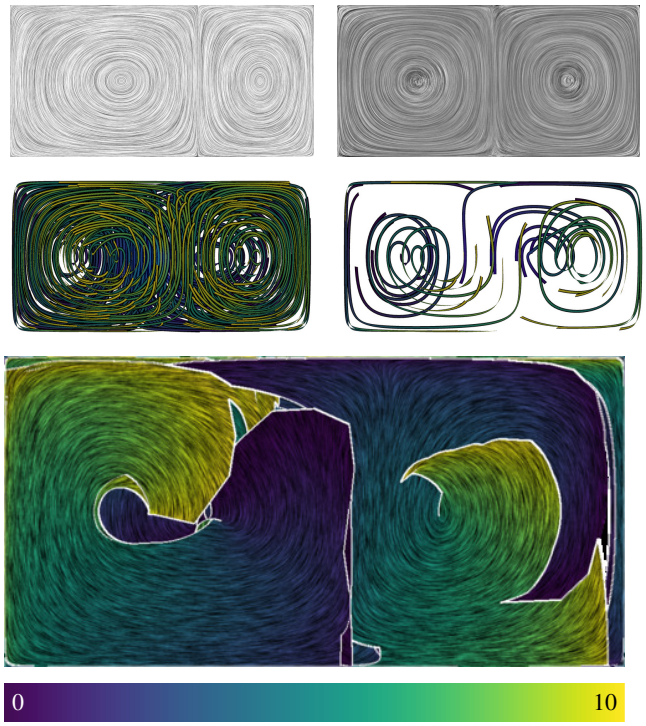

Figure 6: DOUBLE GYRE. *Top left: Standard LIC at time $t = 2.5$. Top right: UFLIC [SK97] with start time $t_0 = 0$ and integration length $\tau = 1$. Center row: View-dependent reduction of random path lines [MCHM10] (left: 100%, right: 10%). Bottom: Steadification with color-coding of time t (21×11 seed grid, $\kappa_{ref} = 15$, $c = 2$).*

Steadification vs. animation Flow steadification does not aim at replacing animation based explorations of unsteady flows. Instead, it complements animations for cases where interactive exploration is not available, such as static visualizations in books, papers, and posters, or, e.g., thumbnail-like previews. With this, flow steadification aims towards a confirmatory analysis and presentation rather than an explorative analysis of flow data [AMST11].

Acknowledgments This work was partially supported by DFG TH 692/14-1 grant.

References

[AMST11] AIGNER W., MIKSCH S., SCHUMANN H., TOMINSKI C.: *Visualization of Time-Oriented Data*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[BBDW17] BECK F., BURCH M., DIEHL S., WEISKOPF D.: A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum* (2017).

[BGH01] BISCHI G.I. L. M., HAUSER H.: Studying basin bifurcations in nonlinear triopoly games by using 3d visualization. *Nonlinear Analysis* (2001).

[Chv79] CHVATAL V.: A greedy heuristic for the set-covering problem. *Math. Oper. Res.* (1979).

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proc. SIGGRAPH* (1993).

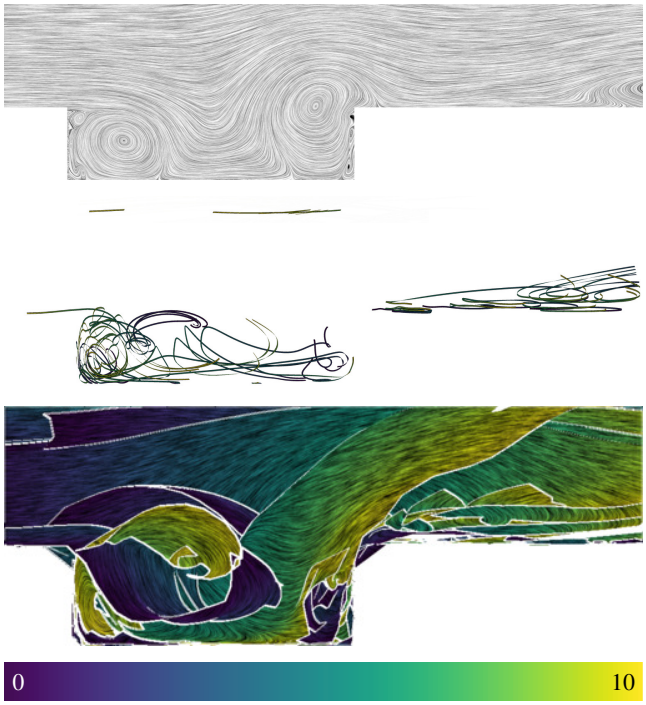

Figure 7: CAVITY. *Top: Standard LIC at time $t = 5$. Center: View-dependent reduction of random path lines using [MCHM10] (10%). Bottom: Steadification with color-coding of t 36×10 seed grid. (51×19 seed grid, $\kappa_{ref} = 10$, $c = 2$)*

[CSD03] CARABALLO E., SAMIMY M., DEBONIS J.: Low dimensional modeling of flow for closed-loop flow control. *AIAA 2003-59* (2003).

[CSFP12] CARNECKY R., SCHINDLER B., FUCHS R., PEIKERT R.: Multi-layer illustrative dense flow visualization. *Comput. Graph. Forum* (2012).

[FC95] FORSSELL L. K., COHEN S. D.: Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics* (1995).

[For94] FORSSELL L. K.: Visualizing flow over curvilinear grid surfaces using line integral convolution. In *Proc. IEEE Visualization* (1994).

[GRT14] GÜNTHER T., RÖSSL C., THEISEL H.: Hierarchical opacity optimization for sets of 3d line fields. *Computer Graphics Forum (Proc. Eurographics)* (2014).

[Hal01] HALLER G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Phys. D* (2001).

[Hal15] HALLER G.: Lagrangian coherent structures. *Annual Review of Fluid Mechanics* (2015).

[HSJW14] HLAWATSCH M., SADLO F., JANG H., WEISKOPF D.: Pathline glyphs. *Comput. Graph. Forum* (2014).

[HY00] HALLER G., YUAN G.-C.: Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D: Nonlinear Phenomena* (2000).

[JEH00] JOBARD B., ERLEBACHER G., HUSSAINI M. Y.: Hardware-accelerated texture advection for unsteady flow visualization. In *Proc. IEEE Visualization* (2000).

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

[JL00] JOBARD B., LEFER W.: Unsteady Flow Visualization by Animating Evenly-Spaced Streamlines. *Computer Graphics Forum* (2000).

[Kar72] KARP R. M.: Reducibility among combinatorial problems. In *Complexity of Computer Computations* (1972).

[KW19] KÖPP W., WEINKAUF T.: Temporal treemaps: Static visualization of evolving trees. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE VIS)* (2019).

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum* (2004).

[LL99] LEEUW W. D., LIERE R. V.: Spotting structure in complex time dependent flow. In *Dagstuhl '97, Scientific Visualization* (1999), IEEE Computer Society.

[LMI02] LIU Z., MOORHEAD II R. J.: AUFLIC: An accelerated algorithm for unsteady flow line integral convolution. In *Proceedings of the symposium on Data Visualisation* (2002).

[LTH06] LI G.-S., TRICOCHE X., HANSEN C.: Gpuflic: Interactive and accurate dense visualization of unsteady flows. In *Proc. EuroVis* (2006).

[LWSH04] LARAMEE R., WEISKOPF D., SCHNEIDER J., HAUSER H.: Investigating swirl and tumble flow with a comparison of visualization techniques. In *Proc. IEEE Visualization* (2004).

[LWW*13] LIU S., WU Y., WE E., LIU M., LIU Y.: Storyflow: Tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics* (2013).

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

[MET*15] MCLOUGHLIN T., EDMUNDS M., TONG C., S R., MASTERS I., CHEN G., MAX N., YEH H.: Visualization of input parameters for stream and pathline seeding. *International Journal of Advanced Computer Science and Applications* (2015).

[MSRT13] MARTINEZ ESTURO J., SCHULZE M., RÖSSL C., THEISEL H.: Global selection of stream surfaces. *Computer Graphics Forum (Proc. Eurographics)* (2013).

[SK97] SHEN H.-W., KAO D. L.: UFLIC: a line integral convolution algorithm for visualizing unsteady flows. In *Proc. IEEE Visualization* (1997).

[SLM05] SHADDEN S. C., LEKIEN F., MARSDEN J. E.: Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena* (2005).

[Sun03] SUNDQUIST A.: Dynamic line integral convolution for visualizing streamline evolution. *IEEE Transactions on Visualization and Computer Graphics* (2003).

[TWHS05] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Topological methods for 2d time-dependent vector fields based on stream lines and path lines. *IEEE Transactions on Visualization and Computer Graphics* (2005).

[WEE03] WEISKOPF D., ERLEBACHER G., ERTL T.: A texture-based framework for spacetime-coherent visualization of time-dependent vector fields. In *Proc. IEEE Visualization* (2003).

[Wij02] WIJK J. J. V.: Image based flow visualization. In *Proc. ACM SIGGRAPH* (2002).

[WRT18] WILDE T., RÖSSL C., THEISEL H.: Recirculation surfaces for flow visualization. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization)* (2018).

[WTS12] WEINKAUF T., THEISEL H., SORKINE O.: Cusps of characteristic curves and intersection-aware visualization of path and streak lines. In *Topological Methods in Data Analysis and Visualization II*. 2012.