Stable Feature Flow Fields

Tino Weinkauf, Holger Theisel, Allen Van Gelder, Alex Pang

Abstract—Feature Flow Fields are a well-accepted approach for extracting and tracking features. In particular, they are often used to track critical points in time-dependent vector fields and to extract and track vortex core lines. The general idea is to extract the feature or its temporal evolution using a stream line integration in a derived vector field – the so-called Feature Flow Field (FFF). Hence, the desired feature line is a stream line of the FFF. As we will carefully analyze in this paper, the stream lines around this feature line may diverge from it. This creates an unstable situation: if the integration moves slightly off the feature line due to numerical errors, then it will be captured by the diverging neighborhood and carried away from the real feature line. The goal of this paper is to define a new FFF with the guarantee that the neighborhood of a feature line has always converging behavior. This way, we have an automatic correction of numerical errors: if the integration moves slightly off the feature line, it automatically moves back to it during the ongoing integration. This yields results which are an order of magnitude more accurate than the results from previous schemes. We present new stable FFF formulations for the main applications of tracking critical points and solving the Parallel Vectors operator. We apply our method to a number of data sets.

1 Introduction

Feature Flow Fields (in the following abbreviated FFF) are a common approach to extract and track different kinds of local features in scalar, vector, and tensor fields. After their introduction in [22], FFF have been applied to track critical points in time-dependent vector fields, extract extremal structures in scalar fields, extract features described by the parallel vectors operator, or to extract degenerate lines in 3D symmetric tensor fields.

The main idea of FFF is to derive a new vector field $\mathbf{f}$ out of the given (scalar, vector, or tensor) field such that the desired features are stream objects of $\mathbf{f}$. Then the features can be extracted in a two-step approach. Firstly, an appropriate set of seeding structures has to be found, then secondly the features are obtained by a numerical stream object integration starting from the seeding structures. Although introduced in a more general context, FFF are mostly used to extract line features. In this paper, we focus on line features as well.

FFF allow to extract and track a large variety of features in scalar, vector and tensor fields [28]. It is attractive for these reasons:

- numerical algorithms for finding seed points and integrating stream lines are well-established,
- the method is independent of the underlying grid structure of the data,
- and FFF can also be used to find and classify bifurcations in the evolution of the features [21].

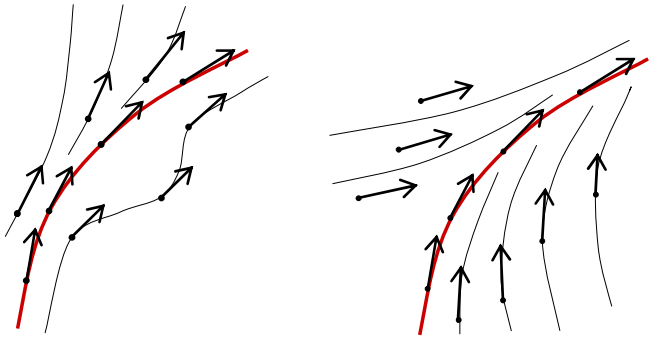
However, numerical instabilities have been reported about FFF [27], [12]. At a rather abstract level they are due to the fact that most FFF formulations use first order derivatives of the original fields, and that the FFF approach therefore consists of a subsequent derivation and integration of the original field which is potentially prone to numerical errors. On a more concrete level, there are two reasons for the instability of FFF:

1) A numerical integration starting on a feature line may (due to numerical integration errors) slightly move off the feature, then the integration may get carried away due to a diverging stream line behavior in the neighborhood.
2) Previous FFF formulations are designed to preserve the data value along any stream line in the whole domain and not only on the feature line. Once the integration is off the feature line, FFF will try to preserve this error and does not attempt to return to the feature.

Note that reason 1 generally holds for any numerical stream line integration, while reason 2 is specific for the FFF integration. A first solution to this instability problem has been presented in [27] which is based on an alternating stream line integration and root finding to enforce a return to the original feature line. This way, [27] is a predictor-corrector approach.

Numerical integration errors cannot be avoided unless the field is trivial. However, for the case of FFF integration they can be automatically corrected. In this paper we introduce a new approach to deal with the instability problem of FFF. As a modification of the FFF formulation we define a new FFF $\mathbf{h}$ such that the feature lines are not only stream lines of $\mathbf{h}$ but that a stream line of $\mathbf{h}$ starting in a neighborhood of the feature line converges to it. This ensures that small numerical integration errors are automatically corrected during the ongoing integration. Figure 1 illustrates the main idea of the paper. Note that we do not use a predictor-corrector approach here. Instead, only a numerical stream line

- *Tino Weinkauf is with Courant Institute of Mathematical Sciences, New York University, E-mail: weinkauf@courant.nyu.edu.*
- *Holger Theisel is head of the Visual Computing Group at University of Magdeburg, E-mail: theisel@isg.cs.uni-magdeburg.de.*
- *Allen Van Gelder is with UC Santa Cruz, Web: http://users.soe.ucsc.edu/~avg/.*
- *Alex Pang is with UC Santa Cruz, E-mail: pang@cse.ucsc.edu.*



(a) Original FFF approach with *diverging* stream lines.

(b) Stable FFF approach guarantees *converging* flow behavior.

Fig. 1. The same feature line (red) encoded as integral curve in two different Feature Flow Fields. With the original approach, starting the integration slightly off the feature may move away from it. Our new stable FFF approach brings the integration back to the feature line.

integration is necessary.

In this paper we establish the concept of stable FFF only for the two most common FFF approaches: tracking critical points in time dependent vector fields, and extracting parallel vector lines.

The rest of the paper is organized as follows: section 2 collects related work. Section 3 analyzes the converging/diverging behavior of stream lines in non-critical points and gives a formal definition of stable FFF. Section 4 introduces the new FFF for tracking critical points in 2D time-dependent vector fields. Section 5 does so for extracting parallel vector lines in 3D fields. An overview of the complete extraction pipeline is given in section 6. Section 7 applies the techniques to different data sets, while conclusions are drawn in section 8.

2 Related Work

Topological methods have been introduced as a visualization tool by Helman and Hesselink [11], where first order critical points are classified by an eigenvalue/eigenvector analysis of the Jacobian matrix. Later, topological methods have been extended to 3D [10], higher order critical points [19], [31], and closed separatrices [32], [5]. Overviews of feature-based flow visualization techniques are given in [17], [13].

For parameter-dependent vector fields, one aims at capturing the evolution of the topological structures. An important class of such fields are time-dependent flows. Tricoche et al. [25] track the location of critical points over time and detect local bifurcations like fold and Hopf bifurcations. This approach works on a piecewise linear 2D vector field and computes and connects the critical points on the faces of a prism cell structure, which is constructed from the underlying triangular grid. An extension to 3D has been given by Garth et al. [9] together with a novel visualization of the paths of the critical points that uses principal component analysis to plot these four-dimensional paths in two dimensions.

Theisel and Seidel [22] introduced an alternative feature tracking approach to the visualization community called *Feature Flow Fields* (FFF). It allows to track a feature by means of a simple stream line integration. FFF were applied to track critical points in time-dependent [23], [7] and two-parameter-dependent vector fields [30]. Furthermore, FFF can be used to extract and track lines defined by the parallel vectors operator [21], which has been introduced by Peikert and Roth [16] as a general feature extraction tool to describe different features such as vortex core lines [29] and ridge lines [26]. FFF have been applied to extract degenerate lines in 3D symmetric tensor fields [34]. Klein and Ertl [12] use FFF to track critical points in scale space.

Tracking critical points in 2D time-dependent vector fields is equivalent to finding the intersection of two 3D isosurfaces as we will see in section 4. Various approaches for intersecting such surfaces were surveyed in [15]. However, numerous difficulties regarding these techniques have been identified in this survey as well. Also note that not all FFF problems can be described as an intersection of surfaces, e.g., extracting vortex core lines as treated in section 5.

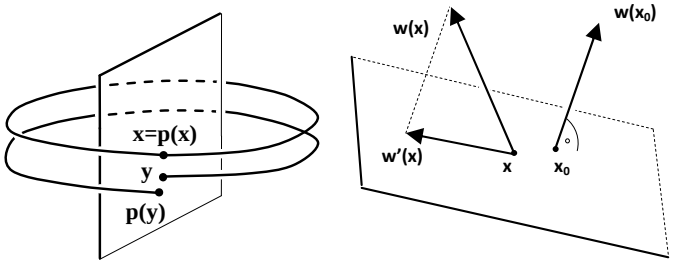
The instability problem of FFF has been addressed for the first time by Van Gelder and Pang [27] in the context of tracking parallel vector lines. The solution presented there is a predictor-corrector approach: after every integration step, the particle is moved back to the feature line by applying a root finding in 3D. This has two consequences: the additional root finding requires substantial computing time, and one major FFF advantage is lost, namely reducing feature tracking to a simple stream line integration. In the following we show how to stabilize the FFF by introducing a new FFF in a closed form. This way, we can use any off-the-shelf numerical integrator instead of a predictor-corrector approach.

Addressing issues such as numerical instabilities, noise, or uncertainty during analysis and visualization is an active field of research. For example, visualizations may be augmented with uncertainty indicators such as glyphs [14] or blur in texture-based visualizations [2]. Several topological methods allow to simplify the topological skeleton. The general idea is to reduce the noise-induced wealth of topological structures to a much smaller set that still exhibits the most relevant structures. Different approaches exist for scalar [8], [3] and vector fields [24], [31], [18]. Chen et al. [6] give a robust method for the extraction of the topology of vector fields that addresses the issue of numerical instabilities by computing the Morse decomposition instead of relying on the computation of individual trajectories.

3 Motivation

We start with a formal definition of Feature Flow Fields.

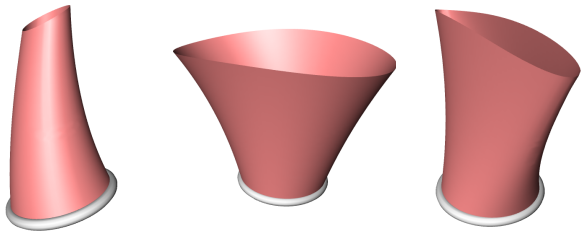
Definition 1 (Feature Flow Field): Let F be a (scalar, vector, or tensor) field, and let L be a set of feature lines



(a) Poincaré map $\mathbf{p}$ with the fix point $\mathbf{x} = \mathbf{p}(\mathbf{x})$ corresponding to the closed stream line. $\mathbf{p}$ describes a discrete dynamical system on the plane which can have attracting, repelling, or saddle behavior.

(b) Projecting a 3D vector field $\mathbf{w}$ onto a plane in order to get the local stream line behavior at $\mathbf{x}_0$.

Fig. 2. Analyzing the global and local behavior of stream lines in a 3D vector field.



(a) Attracting. (b) Repelling. (c) Saddle-like.

Fig. 3. Local behavior of stream lines around a non-critical point $\mathbf{x}_0$ (images from [23]).

(of the same type) in F. The vector field $\mathbf{f}$ is a *Feature Flow Field (FFF)* if all lines of L are stream lines in $\mathbf{f}$.

In this paper we will focus on three-dimensional FFF. In order to make statements about the (in)stability of a stream line integration in such fields, we give a formal description of converging or diverging stream lines in the following. This will lead us to a formal definition of stable FFF in section 3.2.

3.1 Converging and Diverging Stream Lines

The behavior of stream lines can be discussed in a global or a local manner. For a global treatment, note that the feature curves are usually closed curves (unless they hit the boundary of the domain). Therefore, the FFF $\mathbf{f}$ to extract them has a closed stream line. Such lines in 3D vector fields have an attracting, repelling, or saddle behavior which can be obtained by considering the Poincaré map on a plane intersecting the closed stream line [1]. Figure 2a illustrates this. Obviously, the analysis of the Poincaré map needs a global analysis of the field. However, we can introduce a local measure which describes the converging and diverging behavior of a vector field.

Given a 3D vector field $\mathbf{w}$, the analysis of the local converging/diverging behavior around a critical point is well-understood by doing an eigenanalysis of the Jacobian. Here we are interested in a non-critical point $\mathbf{x}_0$. In fact, we want to analyze the behavior of the stream lines passing through a small neighborhood of $\mathbf{x}_0$: they may converge or diverge to/from the stream line through $\mathbf{x}_0$. To get the local stream line behavior of $\mathbf{w}$ in $\mathbf{x}_0$, we consider a plane π through $\mathbf{x}_0$ perpendicular to $\mathbf{w}(\mathbf{x}_0)$. We construct a 2D vector field on π in the following way: for every point $\mathbf{x} \in \pi$, we project $\mathbf{w}(\mathbf{x})$ into π and obtain a 2D vector field $\tilde{\mathbf{w}}$. Figure 2b illustrates this. The field $\tilde{\mathbf{w}}$ has a critical point in $\mathbf{x}_0$. By computing the two eigenvalues β_1, β_2 of its Jacobian, we can characterize the local converging/diverging behavior around $\mathbf{x}_0$. A similar construction has already been applied in [23] in the context of 2D time dependent path lines. Note that β_i do not describe geometric properties of $\mathbf{w}$: any positive scaling of $\mathbf{w}$ will change the values of β_i, but not their signs and therefore not the local stream line behavior.

In order to describe geometric properties of the stream lines around $\mathbf{x}_0$, we consider the Jacobian of the normalized vector field $\overline{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$. The Jacobian $\nabla\overline{\mathbf{w}}$ describes the local changes of the stream lines without considering their acceleration. The eigenvalues of $\nabla\overline{\mathbf{w}}$ are $0, \lambda_1, \lambda_2$, where both λ_i are in close relation to the β_i obtained by the projection method described above: $\beta_i = \|\mathbf{w}\|\lambda_i$. In other words, β_i and λ_i have the same sign and therefore both lead to the same classification of the behavior of stream lines around a non-critical point in the domain (similar to a 2D critical point):

$$\begin{aligned} &\text{repelling behavior:} && 0 < Re(\lambda_1) \leq Re(\lambda_2) \\ &\text{saddle-like behavior:} && Re(\lambda_1) < 0 < Re(\lambda_2) \\ &\text{attracting behavior:} && Re(\lambda_1) \leq Re(\lambda_2) < 0. \end{aligned}$$

Furthermore, the two eigenvectors of $\nabla\overline{\mathbf{w}}$ corresponding to λ_i lie in the plane π that is orthogonal to $\mathbf{w}$, i.e., the same plane that has been used to compute β_i by projection as described above. The proof of these properties of $\nabla\overline{\mathbf{w}}$ is a straightforward exercise in algebra.[1]

For the rest of the paper, we favor λ_i over β_i not only because it allows for a simpler computation, but also since λ_i describe geometric properties of the stream lines of $\mathbf{w}$, i.e., they are independent of the magnitude of $\mathbf{w}$. Figure 3 illustrates the behavior of stream lines around a non-critical point $\mathbf{x}_0$: we integrate a stream surface starting from a small circle in π around $\mathbf{x}_0$ and illustrate attracting, repelling and saddle behavior.

A simple way of computing λ_1, λ_2 is through the following properties, that we will use later:

$$\lambda_1 + \lambda_2 \quad =: \quad s_{\mathbf{w}} = \operatorname{div}\left(\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) \tag{1}$$

$$\lambda_1 \, \lambda_2 \quad =: \quad p_{\mathbf{w}} = \frac{\mathbf{w} \cdot \mathbf{d}_{\mathbf{w}}}{\|\mathbf{w}\|^4} \tag{2}$$

1. We have included a Maple sheet of the proof in the additional materials of the paper.

with

$$\mathbf{d}_{\mathbf{w}} = \begin{pmatrix} \det(\mathbf{w}, \mathbf{w}_y, \mathbf{w}_z) \\ \det(\mathbf{w}_x, \mathbf{w}, \mathbf{w}_z) \\ \det(\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}) \end{pmatrix}. \quad (3)$$

The partial derivatives of $\mathbf{w}$ are denoted as $\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}_z$.

3.2 Stable Feature Flow Fields

Now we are ready to give a formal definition of a stable FFF:

Definition 2 (Stable Feature Flow Field): Let $\mathbf{f}$ be a FFF of a number of feature lines L. $\mathbf{f}$ is *stable* if for every point on a feature line in L the stream lines of $\mathbf{f}$ have an attracting behavior, i.e. $Re(\lambda_1) \leq Re(\lambda_2) < 0$ where λ_1, λ_2 are obtained by (1)–(3) with $\mathbf{w} = \mathbf{f}$.

In a stable FFF $\mathbf{f}$, stream lines of $\mathbf{f}$ adjacent to the feature line converge to it under forward integration. Also note that a stable $\mathbf{f}$ implies an attracting Poincaré map if the feature line is closed.

4 Stable FFF for Tracking Critical Points

In this section we introduce a stable FFF which tracks critical points in 2D time-dependent vector fields. Given such a field

$$\mathbf{v}(x, y, t) = \begin{pmatrix} u(x, y, t) \\ v(x, y, t) \end{pmatrix}, \quad (4)$$

the FFF for tracking critical points as shown in [22] is

$$\mathbf{f}(x, y, t) = (\nabla u)^T \times (\nabla v)^T = \begin{pmatrix} \det(\mathbf{v}_y, \mathbf{v}_t) \\ \det(\mathbf{v}_t, \mathbf{v}_x) \\ \det(\mathbf{v}_x, \mathbf{v}_y) \end{pmatrix}. \quad (5)$$

If the paths of critical points of $\mathbf{v}$ form closed lines in the (x, y, t)-space-time domain (i.e., if they do not touch the domain boundary), the Poincaré map around a feature curve is the identity. This holds because $\mathbf{f}$ as described in (5) is divergence-free since it is the cross product of two gradient fields $\mathbf{f} = (\nabla u)^T \times (\nabla v)^T$. Therefore, the stream lines of $\mathbf{f}$ in a neighborhood of a closed feature line are closed curves as well. However, note that a zero Poincaré map does not prevent us from local integration errors summing up.

In order to get the local converging/diverging behavior of $\mathbf{f}$ at a feature line (i.e., at lines where $\mathbf{v}$ vanishes), we have to check the eigenvalues of $\nabla\overline{\mathbf{f}}$ as described in the previous section by means of $p_{\mathbf{f}} = \frac{\mathbf{f} \cdot \mathbf{d}_{\mathbf{f}}}{\|\mathbf{f}\|^4}$ and $s_{\mathbf{f}} = \operatorname{div} \overline{\mathbf{f}}$ following (1) – (3). Unfortunately, nothing can be said about the signs of the non-zero eigenvalues of $\nabla\overline{\mathbf{f}}$. They can be positive or negative, meaning that $\mathbf{f}$ can act as source, sink or saddle. To overcome this, we introduce a correction vector field

$$\mathbf{g} = \frac{\mathbf{f}}{\|\mathbf{f}\|} \times \begin{pmatrix} \det(\mathbf{v}, \mathbf{v}_x) \\ \det(\mathbf{v}, \mathbf{v}_y) \\ \det(\mathbf{v}, \mathbf{v}_t) \end{pmatrix} \quad (6)$$

which has the following properties:

Property 1: For the corrector field $\mathbf{g}$ described in (6), the following properties hold:

1) Feature lines of $\mathbf{v}$ are critical lines of $\mathbf{g}$.
2) On a feature line, the Jacobian $\nabla\mathbf{g}$ has the eigenvalues $0, -\|\mathbf{f}\|, -\|\mathbf{f}\|$. The eigenvector corresponding to the eigenvalue 0 is $\mathbf{f}$, while the eigenplane corresponding to the other two eigenvalues is perpendicular to $\mathbf{f}$.

The proof of property 1.1 follows directly from (6) and the fact that $\mathbf{v} = \mathbf{0}$ gives $\mathbf{g} = \mathbf{0}$. The proof of property 1.2 is a straightforward exercise in algebra.[1] Note that property 1.2 has the following interpretation: in a small neighborhood of a feature line, $\mathbf{g}$ acts as a sink, meaning that a forward integration of $\mathbf{g}$ converges to the feature line. With this we can define the new FFF

$$\mathbf{h} = \mathbf{f} + \alpha\, \mathbf{g}. \quad (7)$$

Since $\mathbf{g} = \mathbf{0}$ on a feature line, $\mathbf{h}$ is indeed a FFF describing the same feature lines as $\mathbf{f}$. In other words, starting an integration in either $\mathbf{f}$ or $\mathbf{h}$ from the same set of feature seed points (see section 6) yields the same results in theory (no false positives), but the integration in $\mathbf{h}$ is numerically more stable if α is chosen properly.

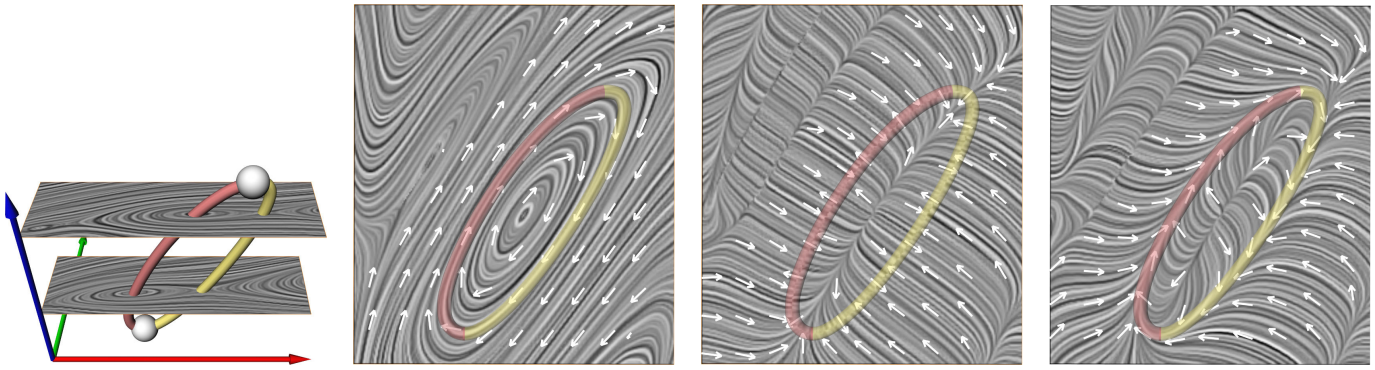
Informally, the role of α in (7) can be described as follows: if α is large enough, then $\mathbf{h}$ is dominated by $\mathbf{g}$, meaning that stream lines of $\mathbf{h}$ are converging towards the feature line. In other words: if α is large enough, $\mathbf{h}$ is a *stable* FFF.

Figure 4 illustrates the vector fields $\mathbf{v}, \mathbf{f}, \mathbf{g}, \mathbf{h}$ for a closeup in the cavity data set (explained in section 7). Here we have a source (red) and a saddle point (yellow) between two fold bifurcations (gray balls), i.e., the critical points appear at the birth fold bifurcation (lower ball) and disappear at the death fold bifurcation. This yields a closed feature line (see figure 4a). The vector fields $\mathbf{f}, \mathbf{g}, \mathbf{h}$ are depicted on a 2D plane in figures 4b - d. As shown in figure 4e, an integration of $\mathbf{f}$ gets carried away from the feature line due to an accumulation of integration errors, which occur especially at the sharp bendings near the fold bifurcations. After some integration time ($t = 40$) the stream line leaves the area completely and runs towards the boundary of the domain. This is clearly not desired. Integrating the feature line in the new FFF $\mathbf{h}$ proves to be stable as shown in figure 4f. This is due to the influence of the correction field $\mathbf{g}$ that changes the neighborhood of the feature line to contain converging stream lines only (see figure 4d).

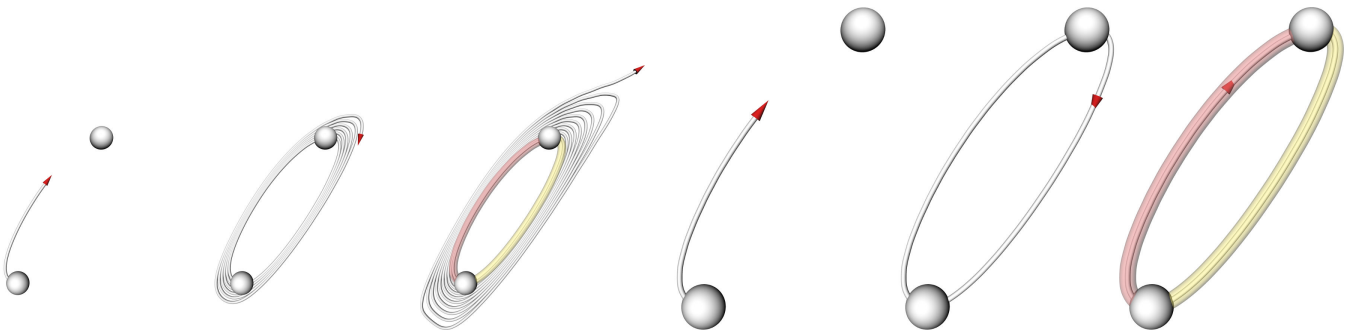
We define α from (7) as a non-constant, smooth scalar field. This way, we can react to the different stream line behaviors in different locations and create a stable FFF that ensures converging behavior everywhere in the domain with a constant strength.

4.1 Obtaining α

In order to find an α that is large enough to ensure a converging behavior, we have to analyze the eigenvalues



(a) Tracked critical points. (b) Original FFF $\mathbf{f}$. (c) Correction vector field $\mathbf{g}$. (d) Stable FFF $\mathbf{h} = \mathbf{f} + \alpha\mathbf{g}$.



(e) Integration in $\mathbf{f}$ at $t = 1$, $t = 20$, and $t = 40$. The integration diverges from the feature line due to locally diverging stream line behavior and accumulated integration errors.

(f) Integration in $\mathbf{h}$ at $t = 1$, $t = 20$, and $t = 40$. The integration stays on the feature line since $\mathbf{h}$ has been designed to have only converging stream line behavior in the proximity of feature lines.

Fig. 4. Closeup of a source (red) and a saddle point (yellow) in the cavity data set. The tracked critical points form lines in the 2D+time diagram shown in a. The blue axis denotes time. The critical points appear at a birth fold bifurcation (lower gray ball) and disappear at a death fold bifurcation (upper gray ball) – thereby forming a closed feature line. The integration of this feature line in the original FFF $\mathbf{f}$ leads eventually away from the actual feature. The integration in the new FFF $\mathbf{h}$ stays on the feature line and is therefore stable. We used a Runge-Kutta scheme of 4th order with adaptive step size control for the integrations.

of $\nabla\overline{\mathbf{h}}$. To do so, we analyze the product $p_{\mathbf{h}}$ and the sum $s_{\mathbf{h}}$ of the two non-zero eigenvalues of $\nabla\overline{\mathbf{h}}$. We get for the partials of $\mathbf{h}$:

$$\begin{aligned} \mathbf{h}_x &= \mathbf{f}_x + \alpha_x\,\mathbf{g} + \alpha\,\mathbf{g}_x \\ \mathbf{h}_y &= \mathbf{f}_y + \alpha_y\,\mathbf{g} + \alpha\,\mathbf{g}_y \\ \mathbf{h}_t &= \mathbf{f}_t + \alpha_t\,\mathbf{g} + \alpha\,\mathbf{g}_t. \end{aligned} \tag{8}$$

On a feature line (i.e., for $\mathbf{v} = \mathbf{0}$) we know $\mathbf{g} = \mathbf{0}$ which simplifies (7) and (8) to

$$\begin{aligned} \mathbf{h} = \mathbf{f} \quad &, \quad \mathbf{h}_x = \mathbf{f}_x + \alpha\,\mathbf{g}_x \\ \mathbf{h}_y = \mathbf{f}_y + \alpha\,\mathbf{g}_y \quad &, \quad \mathbf{h}_t = \mathbf{f}_t + \alpha\,\mathbf{g}_t. \end{aligned} \tag{9}$$

Following equation (3) this gives

$$\begin{aligned} \mathbf{d}_{\mathbf{h}} &= \begin{pmatrix} \det(\mathbf{h}, \mathbf{h}_y, \mathbf{h}_t) \\ \det(\mathbf{h}_x, \mathbf{h}, \mathbf{h}_t) \\ \det(\mathbf{h}_x, \mathbf{h}_y, \mathbf{h}) \end{pmatrix} \\ &= \begin{pmatrix} \det(\mathbf{f}, \mathbf{f}_y, \mathbf{f}_t) \\ \det(\mathbf{f}_x, \mathbf{f}, \mathbf{f}_t) \\ \det(\mathbf{f}_x, \mathbf{f}_y, \mathbf{f}) \end{pmatrix} \\ &\quad + \alpha \begin{pmatrix} \det(\mathbf{f}, \mathbf{f}_y, \mathbf{g}_t) + \det(\mathbf{f}, \mathbf{g}_y, \mathbf{f}_t) \\ \det(\mathbf{f}, \mathbf{f}_t, \mathbf{g}_x) + \det(\mathbf{f}, \mathbf{g}_t, \mathbf{f}_x) \\ \det(\mathbf{f}, \mathbf{f}_x, \mathbf{g}_y) + \det(\mathbf{f}, \mathbf{g}_x, \mathbf{f}_y) \end{pmatrix} \\ &\quad + \alpha^2 \begin{pmatrix} \det(\mathbf{f}, \mathbf{g}_y, \mathbf{g}_t) \\ \det(\mathbf{g}_x, \mathbf{f}, \mathbf{g}_t) \\ \det(\mathbf{g}_x, \mathbf{g}_y, \mathbf{f}) \end{pmatrix}. \end{aligned} \tag{10}$$

From this we get the product of the eigenvalues (cf. equation (2))

$$p_{\mathbf{h}} = \frac{\mathbf{h} \cdot \mathbf{d}_{\mathbf{h}}}{\|\mathbf{h}\|^4} = p_0 + \alpha\, p_1 + \alpha^2 p_2 \tag{11}$$

with

$$p_0 = p_{\mathbf{f}} \tag{12}$$

$$p_1 = \frac{\mathbf{f}}{\|\mathbf{f}\|^4} \cdot \begin{pmatrix} \det(\mathbf{f},\mathbf{f}_y,\mathbf{g}_t)+\det(\mathbf{f},\mathbf{g}_y,\mathbf{f}_t) \\ \det(\mathbf{f},\mathbf{f}_t,\mathbf{g}_x)+\det(\mathbf{f},\mathbf{g}_t,\mathbf{f}_x) \\ \det(\mathbf{f},\mathbf{f}_x,\mathbf{g}_y)+\det(\mathbf{f},\mathbf{g}_x,\mathbf{f}_y) \end{pmatrix} \tag{13}$$

$$p_2 = \frac{\mathbf{f}}{\|\mathbf{f}\|^4} \cdot \begin{pmatrix} \det(\mathbf{f},\mathbf{g}_y,\mathbf{g}_t) \\ \det(\mathbf{g}_x,\mathbf{f},\mathbf{g}_t) \\ \det(\mathbf{g}_x,\mathbf{g}_y,\mathbf{f}) \end{pmatrix}. \tag{14}$$

Furthermore, we get for their sum (cf. equation (1))

$$s_{\mathbf{h}} = \operatorname{div}\left(\frac{\mathbf{h}}{\|\mathbf{h}\|}\right) = s_0 + \alpha\, s_1 \tag{15}$$

with

$$s_0 = \operatorname{div}\left(\frac{\mathbf{f}}{\|\mathbf{f}\|}\right) = s_{\mathbf{f}} \tag{16}$$

$$s_1 = \frac{\operatorname{div}(\mathbf{g})}{\|\mathbf{f}\|}. \tag{17}$$

On a feature line, the following properties hold:

$$s_1 = -2 \quad , \quad s_1^2 - 4\,p_2 = 0 \quad , \quad 2\,s_0\,s_1 - 4\,p_1 = 0. \tag{18}$$

The proof of this is a straightforward exercise in algebra.[1]

For the two non-zero eigenvalues of $\nabla\overline{\mathbf{h}}$ we have

$$\lambda_{1,2}(\alpha) = \frac{s_{\mathbf{h}}}{2} \pm \sqrt{\frac{s_{\mathbf{h}}^2}{4} - p_{\mathbf{h}}} \tag{19}$$

where λ_1, λ_2 with $Re(\lambda_1) \le Re(\lambda_2)$ depend on α. (19) can be rewritten for λ_2 as

$$\lambda_2(\alpha) = \frac{1}{2}\left(s_0 + \alpha\, s_1 + \sqrt{b\,\alpha^2 + c\,\alpha + d}\right) \tag{20}$$

$$\text{with} \quad b = s_1^2 - 4\,p_2\,,\; c = 2\,s_0\,s_1 - 4\,p_1$$
$$d = s_0^2 - 4\,p_0.$$

Inserting (12), (16), (18) into (20) gives

$$\lambda_2(\alpha) = \frac{s_{\mathbf{f}}}{2} - \alpha + \frac{\sqrt{s_{\mathbf{f}}^2 - 4\,p_{\mathbf{f}}}}{2}. \tag{21}$$

In order to obtain α, we introduce a positive parameter $k > 0$ which controls the strength of the attracting behavior of $\mathbf{h}$ along the feature line. In fact, we want to set α such that

$$Re(\lambda_1(\alpha)) \le Re(\lambda_2(\alpha)) = -k < 0. \tag{22}$$

Following definition 2, any $k > 0$ guarantees that $\mathbf{h}$ is a stable FFF, i.e., $\mathbf{h}$ has attracting behavior around its feature lines. The larger the value of k, the more negative are the eigenvalues of $\nabla\overline{\mathbf{h}}$, and the stronger is the attracting behavior of $\mathbf{h}$. Figure 5 illustrates this.

Inserting (21) into (22) gives the solution for α:

$$\alpha = k + \frac{s_{\mathbf{f}}}{2} + \frac{Re\left(\sqrt{s_{\mathbf{f}}^2 - 4\,p_{\mathbf{f}}}\right)}{2}. \tag{23}$$

Note that (23) does not have any numerical problems as long as $\mathbf{f}$ is not vanishing. However, a critical point of $\mathbf{f}$ on a feature line is a structurally unstable feature which we neglect. A critical point of $\mathbf{f}$ outside a feature line is structurally stable, but the choice of $\mathbf{g}$ there does not affect the stable FFF property of $\mathbf{h}$. In case of a vanishing $\mathbf{f}$ outside a feature line we restrict α to a certain predefined upper limit α_{max}. We used $\alpha_{max} = 100$ for all our experiments.

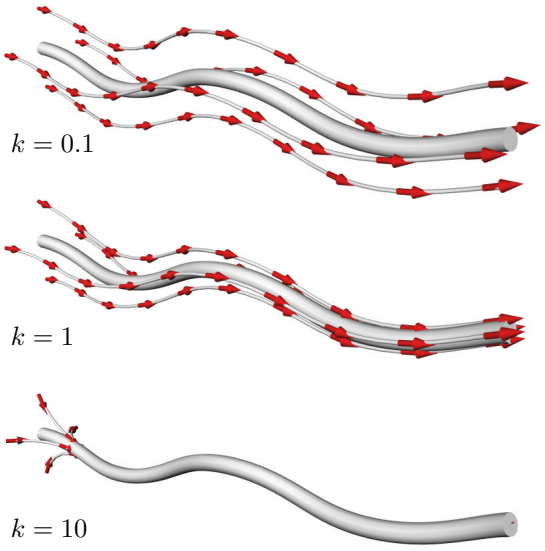


Fig. 5. A feature line (thick) with surrounding stream lines of $\mathbf{h}$. The larger the value of k, the stronger is the attracting behavior of $\mathbf{h}$.

Remark: The α computed in (23) refers to a forward integration of $\mathbf{f}$. If we want to do a stabilized backward integration of $\mathbf{f}$, we do a forward integration of

$$\mathbf{h}' = -\mathbf{f} + \alpha'\,\mathbf{g}, \tag{24}$$

which is a modified version of equation (7) with

$$\alpha' = k + \frac{s_{-\mathbf{f}}}{2} + \frac{Re\left(\sqrt{s_{-\mathbf{f}}^2 - 4\,p_{-\mathbf{f}}}\right)}{2} = k - \frac{s_{\mathbf{f}}}{2} + \frac{Re\left(\sqrt{s_{\mathbf{f}}^2 - 4\,p_{\mathbf{f}}}\right)}{2}. \tag{25}$$

5 Stable FFF for Parallel Vector Lines

It is known that the parallel vectors operator [16] can be extracted using a (non-stable) FFF formulation as given in [21]. In this section, we present a stable version to extract it. Given two 3D vector fields $\mathbf{w}_1, \mathbf{w}_2$, we search for all locations with $\mathbf{w}_1 \parallel \mathbf{w}_2$. The original FFF formulation of [21] defines

$$\mathbf{q} = \mathbf{w}_1 \times \mathbf{w}_2 \tag{26}$$

which gives the FFF

$$\mathbf{f} = \begin{pmatrix} \det(\mathbf{q}_y,\mathbf{q}_z,\mathbf{a}) \\ \det(\mathbf{q}_z,\mathbf{q}_x,\mathbf{a}) \\ \det(\mathbf{q}_x,\mathbf{q}_y,\mathbf{a}) \end{pmatrix} \tag{27}$$

where $\mathbf{a}$ is an almost arbitrary vector field whose choice is discussed in [21]. Note that no statement can be made about the local stability of $\mathbf{f}$: $\nabla\bar{\mathbf{f}}$ may have both positive and negative eigenvalues. Moreover, since (27) is generally not divergence-free, no statements are possible about the Poincaré map around the feature line either.

In order to make $\mathbf{f}$ stable, we use the same approach as in section 4: we define a correction field

$$\mathbf{g} = \frac{\mathbf{f}}{\|\mathbf{f}\|} \times \begin{pmatrix} \det(\mathbf{q}, \mathbf{q}_x, \mathbf{a}) \\ \det(\mathbf{q}, \mathbf{q}_y, \mathbf{a}) \\ \det(\mathbf{q}, \mathbf{q}_z, \mathbf{a}) \end{pmatrix} \tag{28}$$

where $\mathbf{f}$ refers to (27) and $\mathbf{a}$ must be the same field as in (27). Note that $\mathbf{g} = \mathbf{0}$ on a feature line (i.e., for $\mathbf{q} = \mathbf{0}$). Furthermore, property 1.2 from section 4 also holds for the $\mathbf{g}$ in (28).[1]

The new stable FFF $\mathbf{h}$ can be computed by (7) where the choice of the adaptive α is identical to the description in subsection 4.1.

6 Algorithm in a Nutshell

Here we give an overview of the complete extraction pipeline by recapitulating the FFF-based extraction schemes from [23], [21] and collecting our findings from the previous sections. Table 1 accompanies this.

Both feature types (tracked critical points and PV lines) are defined as lines where a vector field becomes zero. The computations start with estimating the first and second derivatives of these fields. Note, that a rough estimate of the second derivative suffices since it only affects the strength of the converging behavior in the neighborhood of the feature line and not the direction of the FFF on the feature line itself, i.e., the second derivative is only needed to compute α. In fact, we used a rather simple estimation with finite differences in our implementation that we compute on the fly during integration.

Based on this we compute $\mathbf{f}, \mathbf{g}, \alpha$ as given in table 1. These are the ingredients for the integration of the stable FFF $\mathbf{h}$. The integration is started at seed points that are extracted as isolated zeros of certain 2D and 3D fields as given in table 1. The choice of these seeds is explained in detail in the literature, e.g. see [23] for the critical point case and [21] for seeding PV lines.

Feature Type	
Tracked Critical Points $\mathbf{v}(x,y,t) = \mathbf{0}$	Parallel Vector Lines $\mathbf{q}(x,y,z) = \mathbf{w}_1 \times \mathbf{w}_2 = \mathbf{0}$
Estimate Derivatives	
$\nabla\mathbf{v}$, $\nabla\nabla\mathbf{v}$	$\nabla\mathbf{q}$, $\nabla\nabla\mathbf{q}$
Compute Vector Fields for Tracking and Stability	
$\mathbf{f} = \begin{pmatrix} \det(\mathbf{v}_y, \mathbf{v}_t) \\ \det(\mathbf{v}_t, \mathbf{v}_x) \\ \det(\mathbf{v}_x, \mathbf{v}_y) \end{pmatrix}$	$\mathbf{f} = \begin{pmatrix} \det(\mathbf{q}_y, \mathbf{q}_z, \mathbf{a}) \\ \det(\mathbf{q}_z, \mathbf{q}_x, \mathbf{a}) \\ \det(\mathbf{q}_x, \mathbf{q}_y, \mathbf{a}) \end{pmatrix}$
$\mathbf{g} = \frac{\mathbf{f}}{\|\mathbf{f}\|} \times \begin{pmatrix} \det(\mathbf{v}, \mathbf{v}_x) \\ \det(\mathbf{v}, \mathbf{v}_y) \\ \det(\mathbf{v}, \mathbf{v}_t) \end{pmatrix}$	$\mathbf{g} = \frac{\mathbf{f}}{\|\mathbf{f}\|} \times \begin{pmatrix} \det(\mathbf{q}, \mathbf{q}_x, \mathbf{a}) \\ \det(\mathbf{q}, \mathbf{q}_y, \mathbf{a}) \\ \det(\mathbf{q}, \mathbf{q}_z, \mathbf{a}) \end{pmatrix}$
Compute Strength of Converging Behavior	
$\alpha = k + \frac{s_\mathbf{f}}{2} + \frac{Re\left(\sqrt{s_\mathbf{f}^2 - 4\,p_\mathbf{f}}\right)}{2}$ with $s_\mathbf{f} = \mathrm{div}\left(\frac{\mathbf{f}}{\|\mathbf{f}\|}\right)$, $p_\mathbf{f} = \frac{\mathbf{f}\cdot\mathbf{d}_\mathbf{f}}{\|\mathbf{f}\|^4}$	
Find Seed Points	
at domain borders: $\mathbf{v} = \mathbf{0}$	at domain borders: $\mathbf{q} = \mathbf{0}$
fold bif.: $\begin{pmatrix} \mathbf{v} \\ \det(\mathbf{v}_x, \mathbf{v}_y) \end{pmatrix} = \mathbf{0}$	inside: $\begin{pmatrix} \mathbf{q} \\ \det(\mathbf{q}_y, \mathbf{q}_z, \mathbf{a}) \end{pmatrix} = \mathbf{0}$
Integrate Feature Lines	
$\mathbf{h} = \mathbf{f} + \alpha\mathbf{g}$	

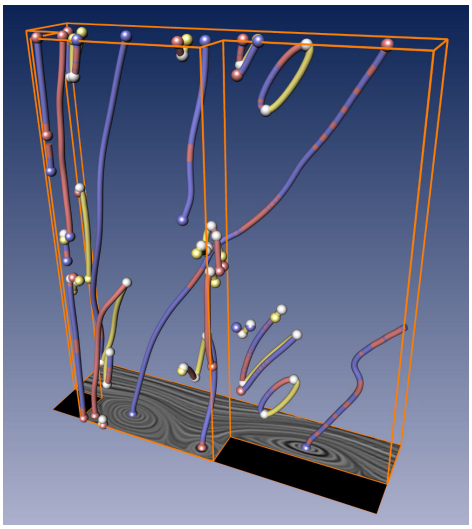
TABLE 1
Overview of the algorithm.

7 Applications

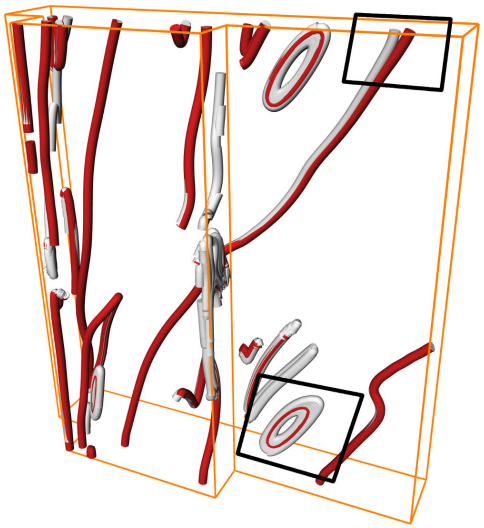
Throughout this paper we use a Runge-Kutta integration scheme of 4th order with adaptive step size control (except for the analysis of the step sizes in figure 6h as detailed below). All computations have been carried out on an AMD64 X2 4400+. The computation times for the integrations in the original FFF and the stable FFF did not differ much – both integrations were finished in under a second for all data sets in this section.

Figure 6 shows an unsteady flow over a 2D cavity. This data set was kindly provided by Mo Samimy and Edgar Caraballo (both Ohio State University) [4] as well as Bernd R. Noack and Ivanka Pelivan (both TU Berlin). 1000 time steps have been simulated using the *compressible* Navier-Stokes equations. It 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 periodic, with a period of about 100 time steps in length, and only the first 100 time steps are shown.

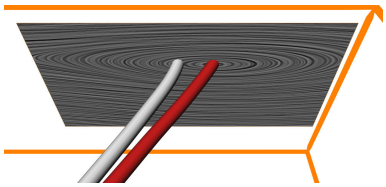
Figure 6a shows the critical points that have been tracked using the new stable FFF. These lines are compared to the results of the original FFF approach in figure 6b. The main differences are highlighted in figures 6c-d. The critical point of figure 6c is a very prominent structure in this data set that persists over all time steps. The tracking of this critical point starts at the first time step and ends at the last one. The results of the original FFF and the stable FFF differ clearly. As elucidated by the LIC plane, the stable FFF managed to keep the integration on the critical point while the original FFF did not. More insight into the integration of this critical point is provided by the diagram in figure 6e. Here we plotted the magnitude $\|\mathbf{v}\|$ over the integration time to measure the error of the extraction process. The error of the stable FFF is clearly below the error of the original



(a) Tracked critical points and fold bifurcations. The spatial domain is denoted by the LIC plane, the third dimension denotes time.

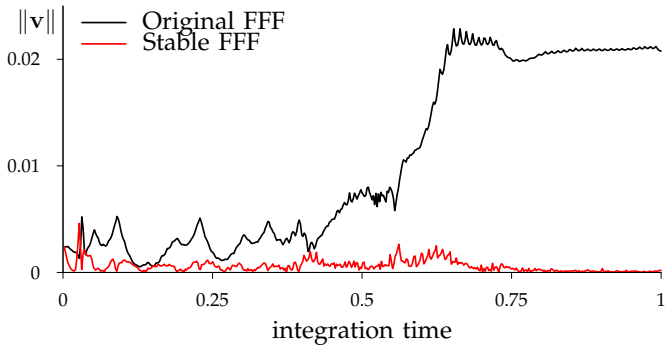


(b) Comparison of the results of the original FFF (gray) and the stable FFF (red).

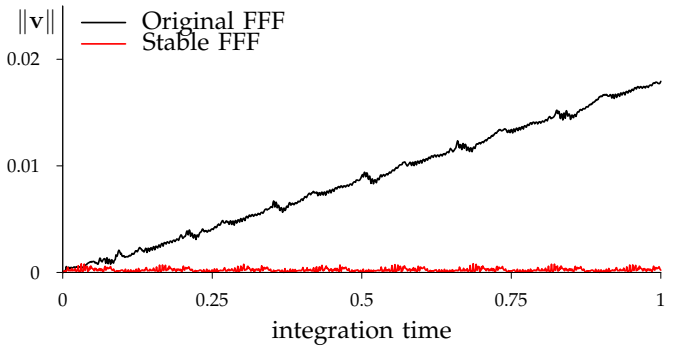


(c) Close-up of 6b (view from below).

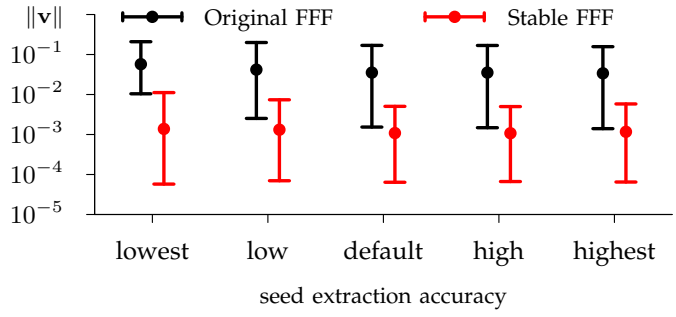
(d) Close-up of 6b. Same structure as in figure 4.



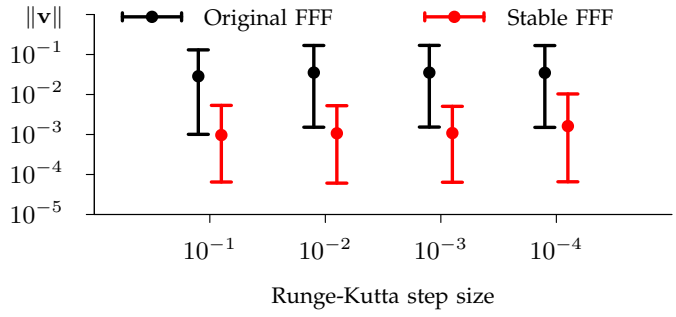
(e) Plot of $\|\mathbf{v}\|$ over integration time for the critical point of 6c.



(f) Plot of $\|\mathbf{v}\|$ over integration time for the critical point of 6d.



(g) Plot of total, minimum and maximum (10%) average of $\|\mathbf{v}\|$ for all critical points over different seed extraction accuracies.



(h) Plot of total, minimum and maximum (10%) average of $\|\mathbf{v}\|$ for all critical points over different step sizes.

Fig. 6. Cavity data set. Topological visualizations and statistical analysis of the results of the original and the stable FFF approaches. The results have been obtained using an adaptive Runge-Kutta scheme of 4th order except for 6h.

FFF. Note, how the stable FFF is able to correct itself as $\|\mathbf{v}\|$ becomes lower towards the end. Figure 6d shows the closed structure already known from figure 4. Here we seeded also from the second fold bifurcation: the original FFF integration diverges towards the inside of the circle. The plot of $\|\mathbf{v}\|$ in figure 6f shows that the original FFF integration is constantly moving away from the critical point.

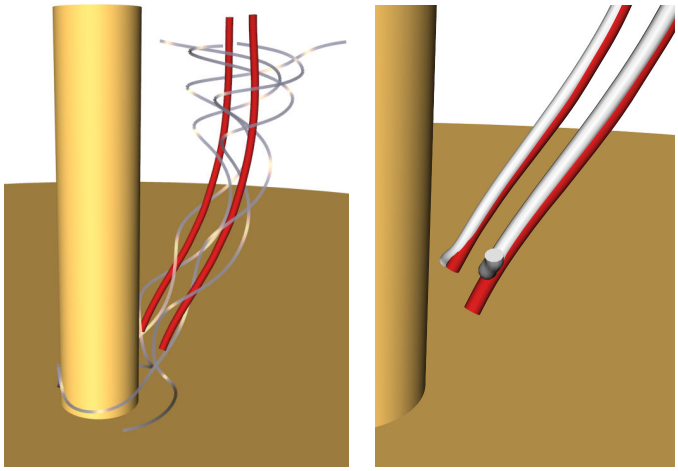
In figures 6g-h we analyze whether and how the integration step size and the seed extraction accuracy influence the outcome of the integrations. To do so, we computed the average of $\|\mathbf{v}\|$ for all tracked critical points and plotted it as a dot. Furthermore, we averaged the highest and lowest 10% of all values of $\|\mathbf{v}\|$ and plotted them as a bracket around the total average to get an estimate of the distribution. Note that the y-axis has a logarithmic scale.

The default seed extraction accuracy matches the resolution of the data set ($256 \times 96 \times 100$). The two coarser/finer resolutions are two and four times lower/higher. It can be seen in figure 6g that the lower resolutions have a negative impact on the quality of the extraction result as the averages of $\|\mathbf{v}\|$ are higher here. Note for the lowest resolution how the original FFF is not able to return to low magnitudes: the minimum average is about 10^{-2} for this case while it is 10^{-3} for the other resolutions. The stable FFF is able to reach the same minimum average of 10^{-4} for all resolutions. Also note that all average values of the stable FFF are at least one order of magnitude below their respective counterparts of the original FFF.

In order to control the step size explicitly we used a 4th order Runge-Kutta scheme with fixed step size and monitored our adaptive integration scheme first. It turns out that the step sizes taken by the adaptive scheme are in the interval $[0.001, 0.02]$. How does the step size influence the quality of the results? For example, would a smaller step size give better results for the original FFF? As evidenced in figure 6h, the choice of the step size does not play a significant role in this setup (as long as it is chosen within reason, of course). However, all average values of the stable FFF are at least one order of magnitude below their respective counterparts of the original FFF. This clearly shows the benefit of using the stable FFF approach over the original one.

In the following we apply our scheme for solving the PV operator to extract vortex core lines in 3D flow fields following the definition of Sujudi/Haimes [20] in the common fashion of Peikert and Roth [16] as lines fulfilling $\mathbf{v}||\mathbf{e}$, where $\mathbf{e}$ is the real eigenvector of $\nabla\mathbf{v}$. In other words, we choose $\mathbf{w}_1 = \mathbf{v}$ and $\mathbf{w}_2 = \mathbf{e}$.

Figure 8 demonstrates the results of our method applied to a flow behind a circular cylinder. The data set was derived by Bernd R. Noack (TU Berlin) from a direct numerical Navier Stokes simulation by Gerd Mutschke (FZ Rossendorf). It resolves the so called 'mode B' of the 3D cylinder wake at a Reynolds number of 300 and a spanwise wavelength of 1 diameter. The data is provided on a $265 \times 337 \times 65$ curvilinear grid as a low-dimensional Galerkin model. The flow exhibits periodic vortex shedding leading to the well-known von Kármán vortex street [33]. This phenomenon plays an important role in many industrial applications, like mixing in heat exchangers or mass flow measurements with vortex counters. However, this vortex shedding can lead to undesirable periodic forces on obstacles, like chimneys, buildings, bridges and submarine towers.



(a) Core line traced using the stable FFF.

(b) Comparison between original (gray) and stable (red) FFF.

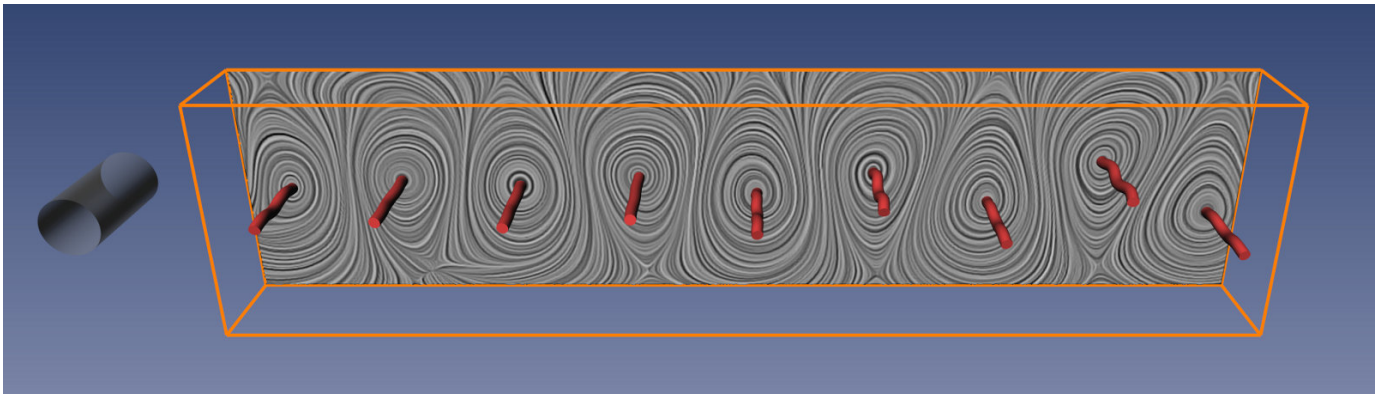
Fig. 7. Vortex structures in the post data set.

Figure 8a shows the vortex core lines that have been integrated using the stable FFF. Figures 8b-c visualize the original and the stable FFF in the proximity of the core lines. The original FFF is almost parallel to the core line. Hence, any inaccuracy in finding the seeds will result in a spatial offset from the real core. With the stable FFF, such inaccuracies are not much of a problem as the stable FFF converges to the real core. Note that figure 8c shows $\mathbf{h}$ for $k = 1$, i.e., the convergence is rather low for illustration purposes. Figure 5 shows $\mathbf{h}$ for $k = 10$, which is the usual setting that we use.

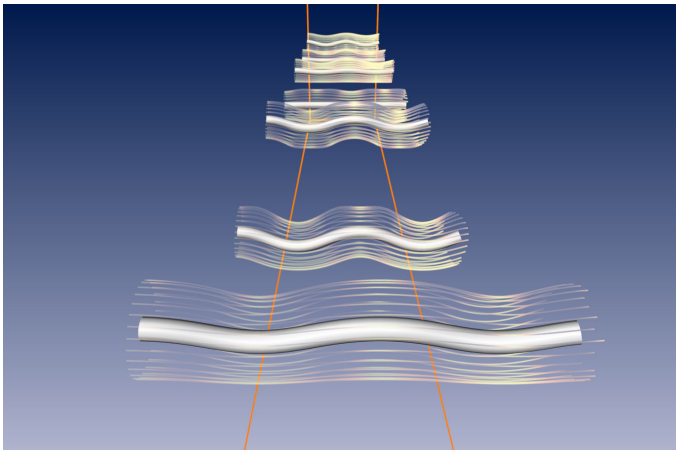
Figure 7 visualizes the vortex core lines in the post data set, which is essentially a cylinder standing on a platform. There are two major vortex core lines in this data set directly behind the cylinder. It has been reported in [27] that this data set poses a problem to the original FFF approach. Indeed, the core line traced in the original FFF bends sharply at the lower end of the core line (figure 7b). The core line traced using the stable FFF does not have this problem. As shown in figure 7a it is in the center of swirling stream line motion.

8 Conclusions

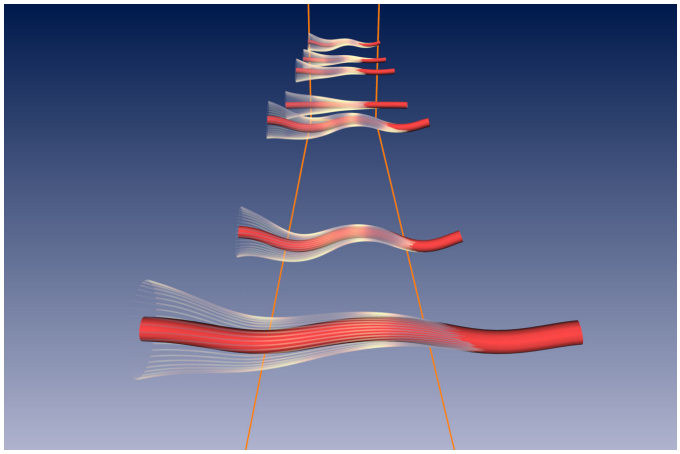
In this paper we introduced the novel notion of a *stable* Feature Flow Field. It extends the common concept of Feature Flow Fields by requiring that the stream lines of the FFF have an attracting behavior in the proximity of a feature line. This stabilizes the integration of the feature



(a) Vortex core line extracted using the stable FFF.



(b) Vortex cores visualized together with stream lines of the original FFF.



(c) Vortex cores visualized together with stream lines of the stable FFF.

Fig. 8. 3D flow behind a cylinder exhibiting the well-known von Kármán vortex street.

line and integration errors are automatically corrected. We developed stable FFF formulations for tracking critical points in 2D time-dependent vector fields and for extracting Parallel Vector lines in 3D vector fields. The results show that in some applications the integrations of the original FFF diverge from the real feature lines, whereas with the new stable FFF the feature lines are captured accurately.

For future work we are interested in developing stable FFF formulations for tracking critical points in 3D time-dependent vector fields as well as tracking vortex core lines over time. Both applications deal with the 4D space-time domain and should bear some similarities.

All numerical computations exhibit a certain amount of error. The shortcoming of the original FFF is to not compensate for this error. With the stable FFF approach we automatically correct errors by always converging to the feature line within a certain region around it. This allows to extract features even in numerically challenging situations using off-the-shelf integrators.

Acknowledgments

Tino Weinkauf is supported by a Feodor Lynen research fellowship of the Alexander von Humboldt foundation. Holger Theisel is partially supported by the SemSeg project under the EU FET-Open grant 226042. The work of Dr. Van Gelder and Alex Pang was partially supported by the UCSC/Los Alamos Institute for Scalable Scientific Data Management (ISSDM).

References

[1] D. Asimov. Notes on the topology of vector fields and flows. Technical report, NASA Ames Research Center, 1993. RNR-93-003.

[2] R.P. Botchen, D. Weiskopf, and T. Ertl. Texture-based visualization of uncertainty in flow fields. In *Proc. IEEE Visualization 2005*, pages 647–654, 2005.

[3] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE TVCG*, 10(4):385 – 396, 2004.

[4] E. Caraballo, M. Samimy, and DeBonis J. Low dimensional modeling of flow for closed-loop flow control. AIAA Paper 2003-0059.

[5] Guoning Chen, Konstantin Mischaikow, Robert S. Laramee, Pawel Pilarczyk, and Eugene Zhang. Vector field editing and periodic orbit extraction using Morse decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):769–785, July - August 2007.

[6] Guoning Chen, Konstantin Mischaikow, Robert S. Laramee, and Eugene Zhang. Efficient Morse decompositions of vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):848–862, 2008.

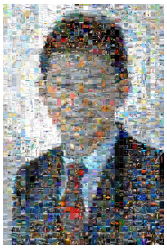
[7] S. Depardon, J. Lasserre, L. Brizzi, and J. Borée. Automated topology classification method for instantaneous velocity fields. *Experiments in Fluids*, 42(5):697–710, May 2007.

[8] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th Sympos. Comput. Geom. 2001*, 2001.
[9] C. Garth, X. Tricoche, and G. Scheuermann. Tracking of vector field singularities in unstructured 3D time-dependent datasets. In *Proc. IEEE Visualization 2004*, pages 329–336, 2004.
[10] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *Proc. IEEE Visualization '91*, pages 33–40, 1991.
[11] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, August 1989.
[12] T. Klein and T. Ertl. Scale-space tracking of critical points in 3D vector fields. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 35–50. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.
[13] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post. Topology-based flow visualization, the state of the art. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 1–19. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.
[14] Alex Pang, Craig M. Wittenbrink, and Suresh K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, November 1997.
[15] N.M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics and Applications*, 13:89–95, 1993.
[16] R. Peikert and M. Roth. The parallel vectors operator - a vector field visualization primitive. In *Proc. IEEE Visualization 99*, pages 263–270, 1999.
[17] Frits H. Post, Benjamin Vrolijk, Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.
[18] J. Reininghaus and I. Hotz. Combinatorial 2D vector field topology extraction and simplification. In *Proc. Topological Methods in Visualization (TopoInVis) 2009*, 2009. to be published.
[19] G. Scheuermann, H. Krüger, M. Menzel, and A. Rockwood. Visualizing non-linear vector field topology. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):109–116, 1998.
[20] D. Sujudi and R. Haimes. Identification of swirling flow in 3D vector fields. Technical report, Department of Aeronautics and Astronautics, MIT, 1995. AIAA Paper 95-1715.
[21] H. Theisel, J. Sahner, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Extraction of parallel vector surfaces in 3d time-dependent fields and application to vortex core line tracking. In *Proc. IEEE Visualization 2005*, pages 631–638, 2005.
[22] H. Theisel and H.-P. Seidel. Feature flow fields. In *Data Visualization 2003. Proc. VisSym 03*, pages 141–148, 2003.
[23] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Topological methods for 2D time-dependent vector fields based on stream lines and path lines. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):383–394, 2005.
[24] X. Tricoche, G. Scheuermann, and H. Hagen. Continuous topology simplification of planar vector fields. In *Proc. Visualization 01*, pages 159 – 166, 2001.
[25] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen. Topology tracking for the visualization of time-dependent two-dimensional flows. *Computers & Graphics*, 26:249–257, 2002.
[26] Xavier Tricoche, Gordon Kindlmann, and Carl-Fredrik Westin. Invariant crease lines for topological and structural analysis of tensor fields. *IEEE Transactions on Visualization and Computer Graphics (Proceedings IEEE Visualization 2008)*, 14(6):1627–1634, 2008.
[27] A. Van Gelder and A. Pang. Using PVsolve to analyze and locate positions of parallel vectors. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):682–695, 2009.
[28] T. Weinkauf. *Extraction of Topological Structures in 2D and 3D Vector Fields*. PhD thesis, University Magdeburg, 2008.
[29] T. Weinkauf, J. Sahner, H. Theisel, and H.-C. Hege. Cores of swirling particle motion in unsteady flows. *IEEE Transactions on Visualization and Computer Graphics (Proceedings IEEE Visualization 2007)*, 13(6):1759–1766, November – December 2007.
[30] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological structures in two-parameter-dependent 2D vector fields. *Computer Graphics Forum*, 25(3):607–616, September 2006. Eurographics 2006, Vienna, Austria, September 04 - 08.
[31] T. Weinkauf, H. Theisel, K. Shi, H.-C. Hege, and H.-P. Seidel. Extracting higher order critical points and topological simplification of 3D vector fields. In *Proc. IEEE Visualization 2005*, pages 559–566, 2005.
[32] T. Wischgoll and G. Scheuermann. Detection and visualization of closed streamlines in planar flows. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):165–172, 2001.
[33] H.-Q. Zhang, U. Fey, B.R. Noack, M. König, and H. Eckelmann. On the transition of the cylinder wake. *Phys. Fluids*, 7(4):779–795, 1995.
[34] X. Zheng, B. Parlett, and A. Pang. Topological lines in 3D tensor fields and discriminant Hessian factorization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):395–407, 2005.

Tino Weinkauf studied computer science with the focus on computer graphics at the University of Rostock, Germany, where he received his M.S. degree in 2000. Based on his research carried out at the Scientific Visualization department of Zuse Institute Berlin (ZIB) on feature-based analysis and comparison techniques for flow fields he received his awarded PhD in computer science from the University of Magdeburg in 2008. Since 2009 he performs research at the Courant Institute of Mathematical Sciences, New York University, based on a Feodor Lynen research fellowship of the Alexander von Humboldt foundation. His current research interests focus on flow and tensor analysis, geometric modeling, and information visualization.

Holger Theisel received his M.S. (1994), Ph.D. (1996) and habilitation (2001) degrees from the University of Rostock (Germany) where he studied Computer Science (1989 – 1994) and worked as a research and teaching assistant (1995 – 2001). He spent 12 months (1994 – 1995) as a visiting scholar at Arizona State University (USA), and 6 months as a guest lecturer at ICIMAF Havana (Cuba). 2002 – 2006 he was a member of the Computer Graphics group at MPI Informatik Saarbrücken (Germany). 2006 – 2007 he was a professor for Computer Graphics at Bielefeld University (Germany). Since October 2007 he is a professor for Visual Computing at the University of Magdeburg. His research interests focus on flow and volume visualization as well as on CAGD, geometry processing and information visualization.

Allen Van Gelder received the BS degree in mathematics from the Massachusetts Institute of Technology, and the PhD degree in computer science from Stanford University in 1987. He is currently Professor of Computer Science at the University of California, Santa Cruz. He has worked in the areas of computer animation, scientific visualization, algorithms, and logic. He is a member of the IEEE Computer Society.



Alex Pang received the BS degree in industrial engineering from the University of the Philippines, and the MS and PhD degrees in computer science from the University of California at Los Angeles in 1984 and 1990, respectively. He is currently Professor of Computer Science at the University of California, Santa Cruz. He has worked in the areas of comparative and uncertainty visualization and flow and tensor visualization. He is a senior member of the IEEE and a member of the IEEE Computer Society.