A Unified Feature Extraction Architecture

Tino Weinkauf[1], Jan Sahner[1], Holger Theisel[2], Hans-Christian Hege[1], and Hans-Peter Seidel[2]

[1] Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany,
{weinkauf, sahner, hege}@zib.de
[2] MPI Informatik Saarbrücken, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany,
{theisel, hpseidel}@mpi-inf.mpg.de

Summary

We present a unified feature extraction architecture consisting of only three core algorithms that allows to extract and track a rich variety of geometrically defined, local and global features evolving in scalar and vector fields. The architecture builds upon the concepts of *Feature Flow Fields* and *Connectors*, which can be implemented using the three core algorithms *finding zeros*, *integrating* and *intersecting* stream objects. We apply our methods to extract and track the topology and vortex core lines both in steady and unsteady flow fields.

1 Introduction

As the resolution of numerical simulations as well as experimental measurements like PIV have evolved significantly in the last years, the challenge of understanding the intricate structures within their massive result data sets has made automatic feature extraction schemes popular. Exploratory techniques alone do not suffice to analyze massive result data sets. Due to the sheer size of the data they have to be complemented by automatic feature extraction schemes, which give a reliable basis for subsequent manual explorations.

In this paper we focus on the treatment of flow fields. They play a vital role in many research areas. Examples are combustion chambers, turbomachinery and aircraft design in industry as well as visualization and control of blood flow in medicine. For this class of data, topological and vortical structures are among the features of interest. Extraction of those structures helps in understanding processes inherent to the flow. This knowledge is the basis for manipulating those processes in terms of flow control.

While [13] gives an overview on flow visualization techniques focusing on feature extraction approaches, we give a short introduction here. Topological methods have become a standard tool to visualize 2D and 3D vector fields because they offer to represent a complex flow behavior by only a limited number of graphical primitives. [6] and [5] introduced them as a visualization tool by extracting critical points and classifying them into sources, sinks and saddles, and integrating certain stream lines called separatrices from the saddles in the directions of the eigenvectors of

the Jacobian matrix. Later, topological methods have been extended to higher order critical points [16] [28], boundary switch points [3] and curves [27], closed separatrices [32] [23], and saddle connectors [22]. In addition, topological methods have been applied to simplify [3] [25] [30], smooth [31], compress [9] and design [19] vector fields.

While they aim at the segmentation of a vector field into areas of different flow behavior, vortex oriented methods highlight turbulent regions of the flow. Recently some work has been done to link these different areas: [4] [26] employ topological methods to analyze the phenomenon of vortex breakdown. Vortices play a major role due to their wanted or unwanted effects on the flow. In turbomachinery design, vortices reduce efficiency, whereas in burning chambers, vortices have to be controlled to achieve optimal mixing of oxygen and fuel. In aircraft design, vortices can both increase and decrease lift. Algorithms for the treatment of vortical structures can be classified in two major categories:

- *Vortex region detection* is based on scalar quantities that are used to define a vortex as a spatial region where the quantity exhibits a certain value range. We refer to them as *vortex region quantities*. Examples of this are regions of high magnitude of vorticity or negative λ_2-criterion [8]. In general, these measures are Galilean invariant, i.e., they are invariant under adding constant vector fields. This is due to the fact that their computation involves derivatives of the vector field only. Isosurfaces or volume rendering are common approaches for visualizing these quantities, which requires the choice of thresholds and appropriate isovalues or transfer functions. As shown in [14], this can become a difficult task for some settings.
- *Vortex core line extraction* aims at extracting line type features that are regarded as centers of vortices. Different approaches exist. [18] [12] consider lines where the flow exhibits a swirling motion around it. [1] extracts vorticity lines seeded at critical points and corrected towards pressure minima. [15] considers stream lines of zero torsion. All of these approaches include a Galilean variant part, i.e., they depend on a certain frame of reference. In contrast to vortex region detection described above, the extraction of those lines is parameter free in the sense that their definition does not refer to a range of values. This eliminates the need of choosing certain thresholds.

In this paper we present an unified approach to extracting and tracking a variety of flow features. Hereby we define the term *feature* as follows:

- A feature is an n-dimensional geometrical structure embedded into a m-dimensional domain.
- It is located inside the domain of the analyzed data.
- It yields certain "insight" into the data.

Finally, the actual definition of a feature depends on the application. In this paper, we mainly treat topological and vortical structures of flow fields. The paper is organized as follows. Section 2 explains the unified feature extraction architecture, while

sections 3 and 4 treat the main concepts behind it, namely Feature Flow Fields and Connectors. We apply our method in section 5 to a number of data sets and feature definitions. Conclusions are drawn in section 6.

2 Unified Feature Extraction Architecture

Almost every feature can be extracted and tracked using a combination of the following core algorithms:

- Finding zeros
- Integrating stream objects
- Intersecting stream objects

We show in section 3 how the first two algorithms can be combined in the Feature Flow Field approach to extracting and tracking features that are defined locally. The intersection of stream objects becomes necessary, where the features we are interested in have a global nature, like closed stream lines. Here the Connectors approach can be applied (section 4).

We now briefly comment on each of the above algorithms.

2.1 Finding Zeros

We are interested in extracting isolated zeros of functions $f : \mathbf{R}^n \to \mathbf{R}^n$. Several approaches exist, some depending on the interpolation scheme:

- Newton-Raphson: use the first derivative to repeatedly predict a zero [12].
- In piecewise linear fields, e.g. tetrahedral grids, the zeros can be computed explicitely.
- In piecewise trilinear fields, e.g. regular grids, a component wise change-of-sign test is a necessary condition for a zero inside the grid cell. Based on this test, a recursive domain decomposition can be applied to the cell that converges to a zero. This method extends to finding zeros of functions $f : \mathbf{R}^n \to \mathbf{R}^m$, is quick, robust, and easy to implement. So we favor this method over the Newton-Raphson-approach for trilinear fields.

2.2 Integration of Stream Objects

Given a flow field $f : \mathbf{R}^n \to \mathbf{R}^n$ we aim at constructing $m+1$-dimensional stream objects from m-dimensional seeding structures.

- For $m = 0$ we obtain a stream line or integral curve.
- For $m = 1$ we obtain a stream surface by triangulating stream lines seeded equidistantly on the seeding line, see [7] for a thorough treatment of this topic.
- Starting from seeding surfaces, we obtain flow volumes, see [10] for implementation details.

Naively integrating stream objects from a seeding structure might result in passing through the whole dataset for each stream object, a costly undertaking, when the dataset is too large to fit into main memory and the number of stream objects is high. Nevertheless Weinkauf et al. showed in [29] that it is possible for a huge subclass of features to do all stream object integrations by sequentially loading the dataset only once, keeping just two consecutive time steps in memory at a time.

Where section 3 shows that finding zeros and integration of stream objects suffices for finding features that are locally defined, stream object integration becomes necessary if such a definition is not at hand.

2.3 Intersection of Stream Objects

Given a flow field $f : \mathbf{R}^n \rightarrow \mathbf{R}^n$ and two m-dimensional ($m > 1$) stream objects R (integrated in forward direction) and A (integrated in backward direction) we aim at extracting the intersection of R and A, i.e., the $m-1$-dimensional stream object that both R and A share.

- For $m = 2$ we obtain a stream line which lies in both intersecting stream surfaces.
- For $m = 3$ we obtain a stream surface which lies in both flow volumes.

Figure 2 shows an example for $m = 2$, where two stream surfaces share a common stream line.

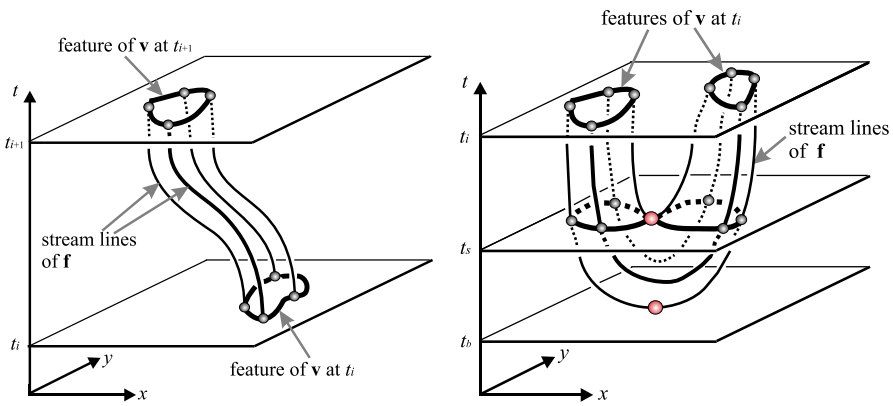
3 Feature Flow Fields

The concept of feature flow fields was first introduced in [21]. It follows a rather generic idea:

Consider an arbitrary point $\mathbf{x}$ known to be part of a feature in a (scalar, vector, tensor) field $\mathbf{v}$. A feature flow field $\mathbf{f}$ is a well-defined vector field at $\mathbf{x}$ pointing into the direction where the feature continues. Thus, starting a stream line integration of $\mathbf{f}$ at $\mathbf{x}$ yields a curve where all points on this curve are part of the same feature as $\mathbf{x}$.

FFF have been used for a number of applications, but mainly for tracking features in time-dependent fields. Here, $\mathbf{f}$ describes the dynamic behavior of the features of $\mathbf{v}$: for a time-dependent field $\mathbf{v}$ with n spatial dimensions, $\mathbf{f}$ is a vector field $\mathbb{R}^{n+1} \rightarrow \mathbb{R}^{n+1}$. The temporal evolution of the features of $\mathbf{v}$ is described by the stream lines of $\mathbf{f}$. In fact, tracking features over time is now carried out by tracing stream lines. The location of a feature at a certain time t_i can be obtained by intersecting the stream lines with the time plane t_i. Figure 1a gives an illustration.

Depending on the dimensionality of the feature at a certain time t_i, the feature tracking corresponds to a stream line, stream surface or even higher-dimensional stream object integration. The stream lines of $\mathbf{f}$ can also be used to detect events of the features:



(a) Tracking features by tracing stream lines. Features at t_{i+1} can be observed by intersecting these stream lines with the time plane $t = t_{i+1}$.

(b) Events: at the time t_b a new feature is born, at the time t_s it splits into two features.

Figure 1 Feature tracking using feature flow fields.

- A birth event occurs at a time t_b, if the feature at this time is only described by one stream line of $\mathbf{f}$, and this stream line touches the plane $t = t_b$ "from above" (i.e., the stream line in a neighborhood of the touching point is in the half-space $t \geq t_b$).
- A split occurs at a time t_s, if one of the stream lines of $\mathbf{f}$ describing the feature touches the plane $t = t_s$ "from above".
- An exit event occurs if all stream lines of $\mathbf{f}$ describing the feature leave the spatial domain.

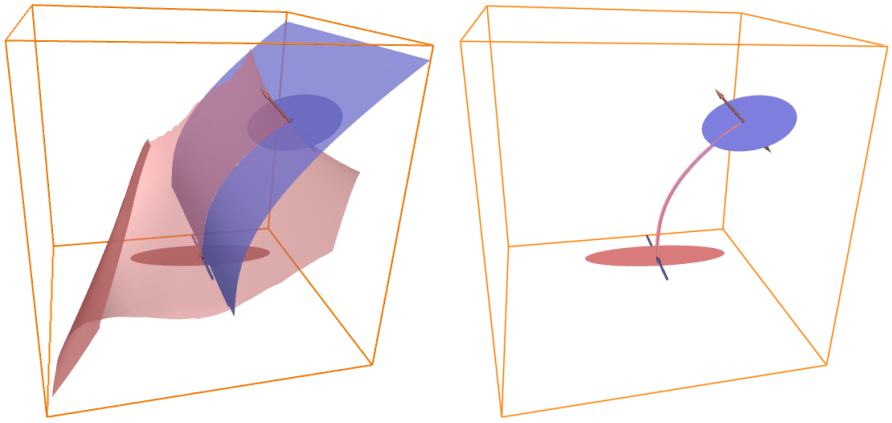
The conditions for the reverse events (death, merge, entry) can be formulated in a similar way. Figure 1b illustrates the different events.

Integrating the stream lines of $\mathbf{f}$ in forward direction does not necessarily mean to move forward in time. In general, those directions are unrelated. The direction in time may even change along the same stream line as it is shown in figure 1b. This situation is always linked to either a birth and a split event, or a merge and a death event.

Even though we treated the concept of FFF in a rather abstract way, we can already formulate the basics of an algorithm to track all occurrences of a certain feature in a time-dependent field:

Algorithm 1 *General FFF-based tracking*

1. *Get seeding points/lines/structures such that the stream object integration of* $\mathbf{f}$ *guarantees to cover all paths of all features of* $\mathbf{v}$.



(a) Two stream surfaces starting from saddle points.

(b) The intersection of the stream surfaces connects both saddles.

Figure 2 Intersection of stream objects.

2. *From the seeding structures: apply a numerical stream object integration of* $\mathbf{f}$ *in both forward and/or backward direction until it leaves the space-time domain.*
3. *If necessary: remove multiply integrated stream objects.*

Algorithm 1 is more or less an abstract template for a specific FFF-based tracking algorithm. Before showing how this template can be used to track critical points and extract and track vortex core lines in flow fields in the applications section 5, we can already note, how the steps of algorithm 1 correlate to the core algorithms given in section 2: the seeding points are usually extracted as critical points of some fields. Then we use the stream object integration from section 2.2 to track the feature.

But what can be done, if the feature of interest does not admit a local definition? Here the connectors approach comes into play.

4 Connectors Approach

Given a flow field $f : \mathbf{R}^n \to \mathbf{R}^n$ and two m-dimensional ($m > 1$) stream objects R (integrated in forward direction) and A (integrated in backward direction) we aim at extracting the intersection of R and A, i.e., the $m-1$-dimensional stream object that both R and A share. Figure 2 shows an example for $m = 2$, where two stream surfaces share a common stream line. Since the intersection of R and A always starts at the repelling seeding structure of R and ends at the attracting seeding structure of A, it is called a *connector*. A connector is a global feature, i.e., it can not be locally defined.

An algorithm for the extraction of line-type connectors has been treated in [22]. To find the intersection between a separation surface in forward integration and a separation surface in backward integration, we integrate both separation surfaces

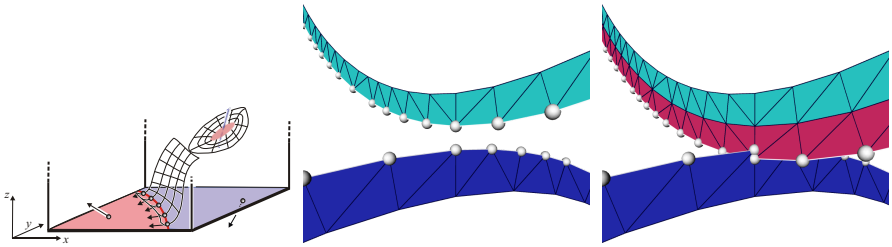


Figure 3 Finding the intersection of two separation surfaces reduced to the problem of intersecting the front triangles of one stream surface with the front line of the other surface.

simultaneously until a first intersection point $\mathbf{p}_1$ is found. After refining this intersection point (see [22] for details), a stream line from $\mathbf{p}_1$ is integrated both forwards and backwards. This stream line is the connector. Figure 3 gives an illustration of this algorithm.

5 Applications

In this section we apply the Feature Extraction Architecture to a variety of feature extraction settings. In section 5.1, topological features are focused while section 5.2 shows how to extract and track vortex core lines using the Feature Extraction Architecture.

5.1 Topological Feature Extraction

Critical points, i.e. isolated points at which the flow vanishes, are perhaps the most important topological feature of vector fields. For static fields, their extraction and classification is well-understood both in the 2D [6] and the 3D case [27]. Critical points also serve as the starting points of certain separatrices, i.e. stream lines/surfaces which divide the field into areas of different flow behavior. Where the direct visualization of those stream surfaces result in cluttered images, Theisel et al. showed in [22] how restricting the display to the intersection lines of those surfaces, called saddle connectors, increases the comprehensibility. This has been achieved by using the connectors approach. In [23] and [24] Theisel et al. showed how to extract and track closed stream lines using the connectors approach.

Considering a stream line oriented topology of time-dependent vector fields, critical points smoothly change their location and orientation over time. In addition, certain bifurcations of critical points may occur. To capture the topological behavior of time-dependent vector fields, it is necessary to capture the temporal behavior of the critical points. Theisel et al. introduced in [21] a FFF-based approach to track critical points, which matches algorithm 1. We now show, how the Feature Extraction Architecture can be applied to this setting.

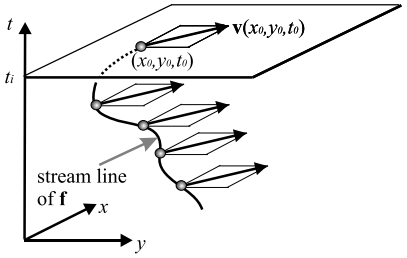


Figure 4 Tracking 2D critical points: all points on a stream line of $\mathbf{f}$ have the same value for $\mathbf{v}$. Note that the depicted stream line is a tangent curve of the feature flow field $\mathbf{f}$ and not of the original velocity field $\mathbf{v}$.

Critical Point Tracking Let $\mathbf{v}$ be a 3D time-dependent vector field, which is given as

$$\mathbf{v}(x,y,z,t) = \begin{pmatrix} u(x,y,z,t) \\ v(x,y,z,t) \\ w(x,y,z,t) \end{pmatrix} \tag{1}$$

in the 4D space-time domain $D = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}] \times [t_{min}, t_{max}]$. We can construct a 4D vector field $\mathbf{f}$ in D with the following properties: for any two points $\mathbf{x}_0$ and $\mathbf{x}_1$ on a stream line of $\mathbf{f}$, it holds $\mathbf{v}(\mathbf{x}_0) = \mathbf{v}(\mathbf{x}_1)$. This means that a stream line of $\mathbf{f}$ connects locations with the same values of $\mathbf{v}$. Figure 4 gives an illustration in 2D. In particular, if $\mathbf{x}_0$ is a critical point in $\mathbf{v}$, then the stream line of $\mathbf{f}$ describes the path of the critical point over time. To get $\mathbf{f}$, we search for the direction in space-time in which both components of $\mathbf{v}$ locally remain constant. This is the direction perpendicular to the gradients of the three components of $\mathbf{v}$:

$$\mathbf{f} \perp \text{grad}(u) = (u_x, u_y, u_z, u_t)^T \;\;,\;\; \mathbf{f} \perp \text{grad}(v) \;\;,\;\; \mathbf{f} \perp \text{grad}(w).$$

This gives a unique solution for $\mathbf{f}$ (except for scaling)

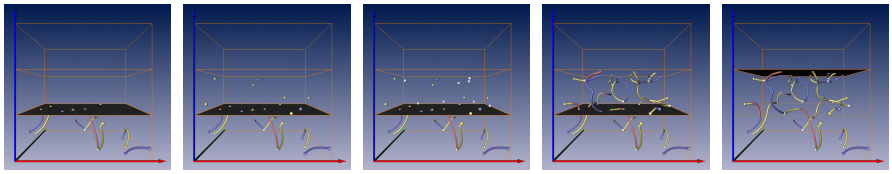
$$\mathbf{f}(x,y,z,t) = \begin{pmatrix} +\det(\mathbf{v}_y, \mathbf{v}_z, \mathbf{v}_t) \\ -\det(\mathbf{v}_z, \mathbf{v}_t, \mathbf{v}_x) \\ +\det(\mathbf{v}_t, \mathbf{v}_x, \mathbf{v}_y) \\ -\det(\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z) \end{pmatrix} \;\; . \tag{2}$$

Theisel et al. showed in [24] that two classes of seeding points guarantee that all paths of critical points are captured: the intersections of the paths with the domain boundaries, i.e. critical points on the boundaries of the space time domain and fold bifurcations, locations where a pair of critical point emerges or vanishes. Fold bifurcations can be characterized by

$$[\; \mathbf{v}(\mathbf{x}) = (0,0,0)^T \;\;,\;\; \det(\mathbf{J}_\mathbf{v}(\mathbf{x})) = 0 \;] \;\; . \tag{3}$$

Applying the Feature Extraction Architecture, we do the following:

– Extraction of seeding structures boils down to finding zeros in the following flow fields: for intersections with the domain boundary, find zeros of the 4 3D



(a) At t_i. (b) Entries. (c) Births. (d) Integration. (e) At t_{i+1}.

Figure 5 Critical points tracked in one sweep through the data by applying the Feature Flow Fields concept.

flow fields

$$\begin{aligned}
\mathbf{v}(x, y, z, t_{min}) = 0 \text{ and } \mathbf{v}(x, y, z, t_{max}) = 0 \text{ for the unknowns } x, y, z,\\
\mathbf{v}(x, y, z_{min}, t) = 0 \text{ and } \mathbf{v}(x, y, z_{max}, t) = 0 \text{ for the unknowns } x, y, t,\\
\mathbf{v}(x, y_{min}, z, t) = 0 \text{ and } \mathbf{v}(x, y_{max}, z, t) = 0 \text{ for the unknowns } x, z, t,\\
\mathbf{v}(x_{min}, y, z, t) = 0 \text{ and } \mathbf{v}(x_{max}, y, z, t) = 0 \text{ for the unknowns } y, z, t.
\end{aligned}$$

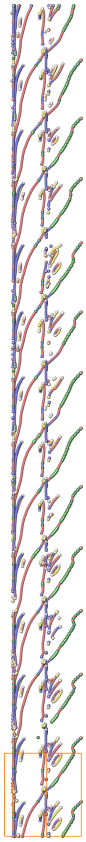
As mentioned above, also the fold bifurcation serve as seeding points. To extract those, a 4D zero extraction has to be applied to formula (3).

– Trace out $\mathbf{f}$ from each of the seeding points to obtain the evolution paths of the critical points. In a postprocessing step remove all lines that are integrated twice, e.g. resulting from stream lines that leave the domain at two different locations.

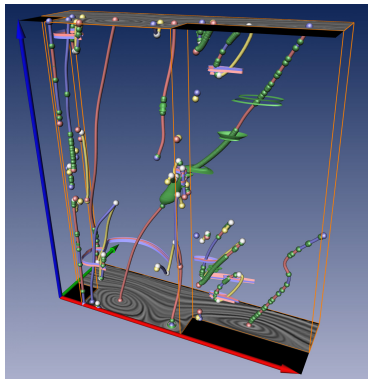
Example: Out-of-core tracking of critical points Weinkauf et al. showed in [29] how to track critical points in 2D and 3D time-dependent vector fields in an effective out-of-core manner: in one sweep and by loading only two slices at once. We applied this algorithm to a random 2D time-dependent data set. Random vector fields are useful tools for a proof-of-concept of topological methods, since they contain a maximal amount of topological information. Figure 5 shows the execution of the tracking algorithm between two consecutive time steps t_i and t_{i+1}.

Example: Cavity Figure 6 shows the visualization of a vector field describing the flow over a 2D cavity. This data set was kindly provided by Mo Samimy and Edgar Caraballo (both Ohio State University) [2] as well as Bernd R. Noack (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 topological structures of the full data set visualized in Figure 6a elucidate the quasi-periodic nature of the flow. Figures 6b-c show approximately one period – 100 time steps – of the full data set, while Figures 6d-e point out some topological details.

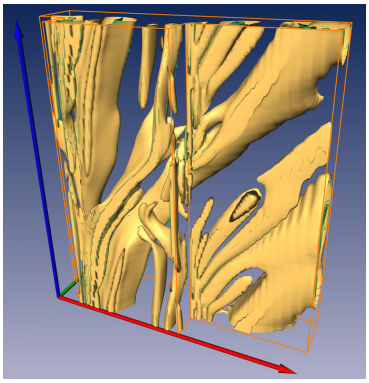
Figures 6b-c both reveal the overall movement of the topological structures – the most dominating ones originating in or near the boundaries of the cavity itself. The



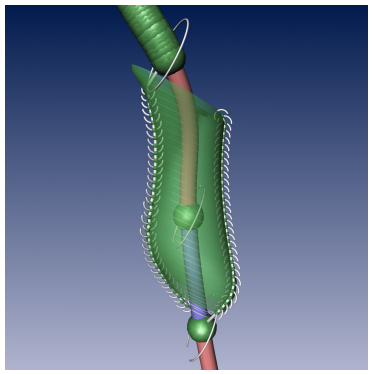
(a) 1000 time steps.



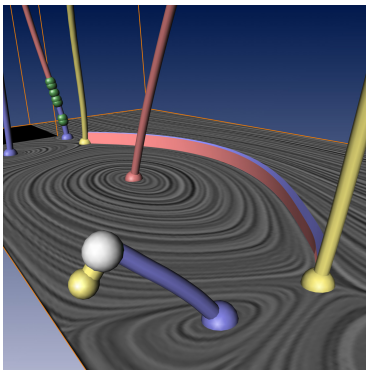
(b) Stream line oriented topology of the first 100 time steps.



(c) Path line oriented topology of the first 100 time steps.



(d) Tracked closed stream line starting and ending in a Hopf bifurcation.



(e) Detail view with a saddle connection and a fold bifurcation.

Figure 6 2D time-dependent flow at a cavity. The datasets consists of 1000 time steps which have been visualized in (a). All other images show the first 100 time steps.

quasi-divergence-free behavior outside the cavity is affirmed by the fact that a high number of Hopf bifurcations has been found in this area. The tracked closed stream line in Figure 6d starts in a Hopf bifurcation and ends in another one – thereby enclosing a third Hopf. Figure 6e shows a detailed view of time step 22, where a saddle connection has been detected. In the front of this figure a sink is going to join and disappear with a saddle, which just happened to enter at the domain boundary.

5.2 Vortex Core Line Extraction And Tracking

We apply the Feature Extraction Architecture to Vortex Core Line Extraction. While [12] gives a good overview of existing vortex core line definitions, we use the most

prominent technique by Sujudi and Haimes [18]. Using the notation of [12] and denoting $\mathbf{w}_1 := \mathbf{v}, \mathbf{w}_2 := \nabla\mathbf{v} \cdot \mathbf{v}$, we define a vortex core line as locations where

$$\mathbf{w}_1 \parallel \mathbf{w}_2, \tag{4}$$

where $\parallel$ denotes vector parallelity and $\mathbf{v}$ is a time dependent flow field as in (1).

In this setting, the Feature Extraction Architecture can solve different tasks:

1. Extract vortex core lines at some time step t_0.
2. Track a given vortex core line in time, i.e., given a vortex core line at some time t_0, compute the evolution path of this vortex core line in the 4D-spacetime domain. This will assemble a surface.
3. Extract the complete vortex core line surface from 2 at once and use it for vortex core line display and tracking in time.

Spatial Extraction of Vortex Core Lines By (4), a point $\mathbf{x}$ is on a vortex core line, whenever

$$\mathbf{s}(x,y,z,t) := \begin{pmatrix} k(x,y,z,t) \\ m(x,y,z,t) \\ n(x,y,z,t) \end{pmatrix} := \mathbf{v} \times \nabla\mathbf{v} \cdot \mathbf{v} = \mathbf{w}_1 \times \mathbf{w}_2 = 0. \tag{5}$$

Given a point $\mathbf{x}_0 = (x_0, y_0, z_0, t_0)^T \in D = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}] \times [t_{min}, t_{max}]$ on a vortex core line (i.e. $\mathbf{s}(\mathbf{x}_0) = 0$), we can trace stream lines of the following feature flow field $\mathbf{f}$ from [20] to extract vortex core lines from the seed point $\mathbf{x}_0$ at time t_0:

$$\mathbf{f}(x,y,z,t_0) = \begin{pmatrix} e \\ f \\ g \end{pmatrix} = \begin{pmatrix} \det(\mathbf{s}_y, \mathbf{s}_z, \mathbf{a}) \\ \det(\mathbf{s}_z, \mathbf{s}_x, \mathbf{a}) \\ \det(\mathbf{s}_x, \mathbf{s}_y, \mathbf{a}) \end{pmatrix}. \tag{6}$$

For the choice of $\mathbf{a}$ we refer to [20].

In the notation of the Feature Extraction Architecture, the complete skeleton of vortex core lines at t_0 can be extracted as follows:

– From [20] we know, that all vortex core lines are either closed or cross the boundary. Therefore, we extract as starting points all intersections of the vortex core lines with the 2D spatial domain boundary $[x_{min}, y, z, t_0] \cup [x_{max}, y, z, t_0] \cup [x, y_{min}, z, t_0] \cup [x, y_{max}, z, t_0] \cup [x, y, z_{min}, t_0] \cup [x, y, z_{max}, t_0] \cup$ at timestep t_0, e.g. at x_{min}:

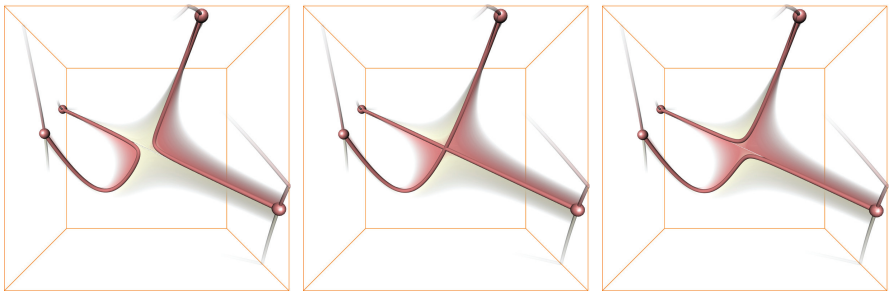
$$\mathbf{s}(x_{min}, y, z, t_0) = (0,0,0)^T. \tag{7}$$

This is a function $\mathbf{R}^2 \to \mathbf{R}^3$ with isolated zeros due to the dependencies of the components in the cross product (5). Closed vortex core lines can be detected by finding isolated zeros in the field

$$[\ \mathbf{s}(\mathbf{x}) = (0,0,0)^T \quad , \quad e(\mathbf{x}) = 0\ ], \tag{8}$$

a function $\mathbf{R}^3 \to \mathbf{R}^4$, see again [20] for details.

– Given those seeding points, we can extract all vortex core lines at time step t_0 by tracing stream lines of $\mathbf{f}$.



(a) Shortly before. (b) The event. (c) Shortly after.

Figure 7 Example of the visualization of vortex core line surfaces. Shown is a saddle bifurcation of vortex core lines. The surfaces are displayed bright for future, dark for past times.

Tracking of Vortex Core Lines in Time For any vortex core line at a given time t_0, [20] shows that stream lines of $\mathbf{g}$ seeded from the vortex core line tracks the temporal evolution of the vortex core line:

$$\mathbf{g}(x,y,z,t) = \begin{pmatrix} \mathbf{h} \times \mathbf{f} \\ \|\mathbf{f}\|^2 \end{pmatrix} = \begin{pmatrix} \mathbf{h} \times \mathbf{f} \\ e^2 + f^2 + g^2 \end{pmatrix} \tag{9}$$

with

$$\mathbf{h}(x,y,z,t) = \begin{pmatrix} \det(\mathbf{s}_x, \mathbf{s}_t, \mathbf{a}) \\ \det(\mathbf{s}_y, \mathbf{s}_t, \mathbf{a}) \\ \det(\mathbf{s}_z, \mathbf{s}_t, \mathbf{a}) \end{pmatrix} \quad . \tag{10}$$

A complete Vortex Core Line Skeleton In the 4D space time domain D, the vortex core lines build surface structures. In [20] a detailed algorithm is given, how this surface structure can be extracted based on a bifurcation analysis of the above feature flow field. In the Feature Extraction Architecture notation, the algorithm reads as follows:

- Compute the seeding structures:
  1. Compute the intersection curves of the vortex core line surface with the spatial boundaries of D. This can be done by spatial extraction of vortex core lines as explained above.
  2. Extract all local bifurcations introduced in [20] by finding zeros of some function $\mathbf{R}^4 \rightarrow \mathbf{R}^4$.
  3. Extract closed vortex core lines at the times $t = t_{min}$ and $t = t_{max}$ respectively, pick a point on each extracted closed line, and apply a stream line integration of $\mathbf{g}$ starting from them.
  4. Start a stream line integration of $\mathbf{g}$ from all inflow boundary bifurcations (zeros of some function $\mathbf{R}^3 \rightarrow \mathbf{R}^3$).
- Extract and visualize the vortex core line surface for a time interval $[t_0, t_1]$ with $t_{min} \leq t_0 \leq t_1 \leq t_{max}$):

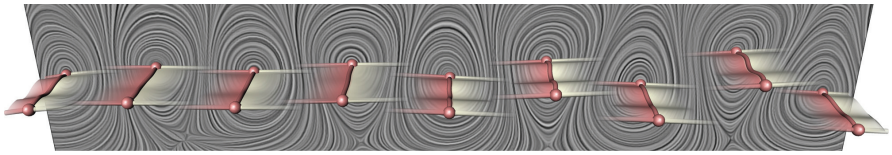


Figure 8 Flow behind a circular cylinder. Shown are vortex core lines in a certain frame of reference. Their evolution over time is tracked by our algorithm and depicted using transparent surfaces. Dark color encodes the past while bright shows the future.

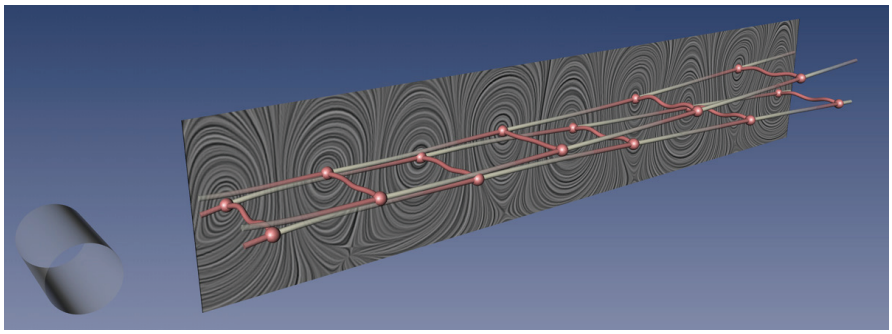


Figure 9 Flow behind a circular cylinder. The extracted seeding lines elucidate the alternating evolution of the vortical structures in transverse direction.

1. Load the seeding lines obtained above.
2. Identify all parts of the seeding lines with t-values between t_0 and t_1.
3. Starting from these seeding lines, apply a stream surface integration of $\mathbf{f}$ until it leaves D or returns to its starting point.
4. Visualize the stream surfaces obtained in 3.

As projecting the complete vortex core surface to space leads to self-intersections already in quite simple settings, we use the following approach to visualize the evolution of vortex core line structures: at a given time we draw the vortex core lines as solid tubes inside the vortex core surface that is displayed only for a certain time range for future and past. At the boundary of the space domain the corresponding seeding lines are given for a larger time interval. Both the surfaces and the seeding lines fade out away from the current time. We use color coding to indicate past (dark) and future (bright). Figure 7 shows the evolution of a specific inner bifurcation called saddle bifurcation. Note that the width of the surface in figures 7a and 7c confirms the intuition that the most drastic movements of the vortex core line over time takes place near the bifurcation points.

Example: Flow behind a Circular Cylinder Figures 8 and 9 demonstrate results of the Unified Feature Extraction Architecture of vortex core line extraction in 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 [11] [33]. The examined time range is $[0, 2\pi]$. The flow exhibits periodic vortex shedding leading to the well known von Kármán vortex street. 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.

6 Conclusions

In this paper we exemplified that a rich variety of flow features can be extracted and tracked by a combination of only three core algorithms, namely *finding zeros*, *integrating* and *intersecting* stream objects. The so-defined *Unified Feature Extraction Architecture* builds upon the concepts of *Feature Flow Fields* and *Connectors*.

Acknowledgments

We thank Bernd R. Noack for the fruitful discussions and providing the Galerkin model of the cylinder data set. All visualizations in this paper have been created using AMIRA – a system for advanced visual data analysis [17] (see http://amira.zib.de/).

References

[1] D.C. Banks and B.A. Singer. A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):151–163, 1995.

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

[3] W. de Leeuw and R. van Liere. Collapsing flow topology using area metrics. In *Proc. IEEE Visualization '99*, pages 149–354, 1999.

[4] 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.

[5] 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.

[6] 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.

[7] J. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proc. IEEE Visualization '92*, pages 171–177, 1992.

[8] J. Jeong and F. Hussain. On the identification of a vortex. *J. Fluid Mechanics*, 285:69–94, 1995.

[9] S.K. Lodha, J.C. Renteria, and K.M. Roskin. Topology preserving compression of 2D vector fields. In *Proc. IEEE Visualization 2000*, pages 343–350, 2000.

[10] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proc. Visualization 93*, pages 19–24, 1993.

[11] B.R. Noack and H. Eckelmann. A low-dimensional galerkin method for the three-dimensional flow around a circular cylinder. *Phys. Fluids*, 6:124–143, 1994.

[12] R. Peikert and M. Roth. The parallel vectors operator - a vector field visualization primitive. In *Proc. IEEE Visualization 99*, pages 263–270, 1999.

[13] F.H. Post, B. Vrolijk, H. Hauser, R.S. Laramee, and H. Doleisch. Feature extraction and visualisation of flow fields. In *Proc. Eurographics 2002, State of the Art Reports*, pages 69–100, 2002.

[14] M. Roth and R. Peikert. Flow visualization for turbomachinery design. In *Proc. Visualization 96*, pages 381–384, 1996.

[15] M. Roth and R. Peikert. A higher-order method for finding vortex core lines. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 143–150, Los Alamitos, 1998. IEEE Computer Society Press.

[16] 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.

[17] D. Stalling, M. Westerhoff, and H.-C. Hege. Amira: A highly interactive system for visual data analysis. *The Visualization Handbook*, pages 749–767, 2005.

[18] 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.

[19] H. Theisel. Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum (Eurographics 2002)*, 21(3):595–604, 2002.

[20] 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.

[21] H. Theisel and H.-P. Seidel. Feature flow fields. In *Data Visualization 2003. Proc. VisSym 03*, pages 141–148, 2003.

[22] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proc. IEEE Visualization 2003*, pages 225–232, 2003.

[23] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Grid-independent detection of closed stream lines in 2D vector fields. In *Proc. Vision, Modeling and Visualization 2004*, 2004.

[24] 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.

[25] X. Tricoche, G. Scheuermann, and H. Hagen. Continuous topology simplification of planar vector fields. In *Proc. Visualization 01*, pages 159 – 166, 2001.

[26] Xavier Tricoche, Christoph Garth, Gordon Kindlmann, Eduard Deines, Gerik Scheuermann, Markus Ruetten, and Charles Hansen. Visualization of intricate flow structures for vortex breakdown analysis. In *Proc. IEEE Visualization 2004*, pages 187–194, 2004.

[27] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Boundary switch connectors for topological visualization of complex 3D vector fields. In *Data Visualization 2004. Proc. VisSym 04*, pages 183–192, 2004.

[28] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological construction and visualization of higher order 3D vector fields. *Computer Graphics Forum (Eurographics 2004)*, 23(3):469–478, 2004.

[29] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Feature flow fields in out-of-core settings. In *Proc. Topo-In-Vis 2005, Budmerice, Slovakia*, 2005.

[30] T. Weinkauf, H. Theisel, K. Shi, H.-C. Hege, and H.-P. Seidel. Topological simplification of 3d vector fields by extracting higher order critical points. In *Proc. IEEE Visualization 2005*, pages 559–566, 2005.

[31] R. Westermann, C. Johnson, and T. Ertl. Topology-preserving smoothing of vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):222–229, 2001.

[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.