Grid-Independent Detection of Closed Stream Lines in 2D Vector Fields

Holger Theisel[1] Tino Weinkauf[2] Hans-Christian Hege[2] Hans-Peter Seidel[1]

[1] MPI Informatik, Saarbrücken, Germany – {theisel,hpseidel}@mpi-sb.mpg.de
[2] Zuse Institute Berlin (ZIB), Germany – {weinkauf,hege}@zib.de

Abstract

We present a new approach to detecting isolated closed stream lines in 2D vector fields. This approach is based on the idea of transforming the 2D vector field into an appropriate 3D vector field such that detecting closed stream lines in 2D is equivalent to intersecting certain stream surfaces in 3D. Contrary to pre-existing methods, our approach does not rely on any underlying grid structure of the vector field. We demonstrate the applicability and stability by applying it to a test data set.

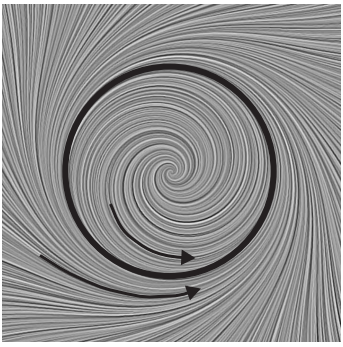


Figure 1: Example vector field with an isolated closed stream line.

1 Introduction

Topological methods are standard tools for the visualization of 2D vector fields. The main idea behind them is to segment the flow into areas of similar flow behavior by extracting critical points and separatrices. Visualizing the topological skeleton is attractive since even a complex flow behavior can be represented by a limited number of graphical primitives.

After the introduction of topological methods to the visualization community in [6], an intensive research has been done in this field. [12] treat higher order critical points, i.e. critical points with a possibly vanishing Jacobian. In [3], separatrices starting from boundary switch points are discussed. Topological methods are used to simplify [3, 4, 20, 21], smooth [26], compress [10] and design [15] vector fields. In [9, 2, 17], topology-based 2D vector field metrics are defined. The topological behavior of time-dependent vector fields is analyzed in [22, 23, 16, 19]. Topological features of 3D vector fields are extracted and visualized in [7, 5, 11, 18, 24, 25].

Most topological features of a 2D vector field can be obtained by a local analysis. For example, to decide whether a point $\mathbf{x}$ is a critical point or boundary switch point in a vector field $\mathbf{v}$, the consideration of $\mathbf{v}$ at $\mathbf{x}$ and its neighborhood is sufficient. Since critical and boundary switch points also act as starting points of certain separatrices, they can also be obtained by a local analysis. However, there are also topological features which can only be detected by a global analysis of $\mathbf{v}$. Here we treat isolated closed stream lines in a vector field. They are important topological features because they separate a vector field into two areas of different flow behavior: inside and outside the closed stream line. Figure 1 shows an example of a simple vector field with an isolated closed stream line. Note that in this example stream lines close to the closed stream line converge to it without actually reaching it.

A first approach to detecting closed stream lines was given in [27] which uses the underlying grid structure of a piecewise linear vector field: each grid cell is analyzed concerning the re-entering behavior of the stream lines starting at its boundaries. Based on this approach, [28] tracks closed stream lines over time by applying a contouring & connecting - like approach: at each time step closed stream lines are detected independently of each other, then the corresponding lines in adjacent time steps are connected. Another approach to tracking closed

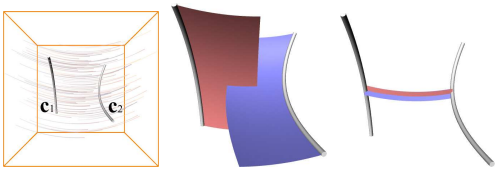
stream lines in 2D time dependent vector fields was presented in [19]. There, a new closed stream line is detected close (in space-time) to an already known one. This process starts at Hopf bifurcations and periodic blue sky bifurcations. However, closed stream lines may also occur in the first time step of the time-dependent data set, and may move over time until the last time step without undergoing any bifurcation. Because of this, it is also necessary to have a stable algorithm which detects closed stream lines for 2D (steady) vector fields.

In this paper we present an alternative approach for detecting closed stream lines. This approach does not rely on any underlying grid and is therefore more general than pre-existing solutions. The main idea is to transform a 2D vector field $\mathbf{v}$ into an appropriate 3D vector field $\mathbf{w}$ such that the search for closed stream lines in $\mathbf{v}$ is equivalent to the search of intersection curves of certain stream surfaces in $\mathbf{w}$. This approach is motivated by the fact that recently robust algorithms to intersect stream surfaces have been proposed ([18]) and applied to extract a number of topological features of 3D ([24]) and 2D time dependent ([19]) vector fields.

The rest of the paper organized as follows: section 2 recollects an approach to intersecting stream surfaces in 3D vector fields. Section 3 describes the core of our approach: how to get a 3D vector field $\mathbf{w}$ from $\mathbf{v}$, and how to find seeding lines in $\mathbf{w}$ such that closed stream lines in $\mathbf{v}$ correspond to intersecting stream surfaces in $\mathbf{w}$. Section 4 describes how to find a system of seeding lines in $\mathbf{w}$ such that all closed stream lines in $\mathbf{v}$ are guaranteed to be detected. Section 5 presents applications of our algorithm while conclusions are drawn in section 6.

2 Intersecting Stream Surfaces

Given a 3D vector field $\mathbf{w}$, a stream surface is uniquely defined by specifying a seeding line $\mathbf{c}$. Then the stream surface can be obtained by applying a numerical stream surface integration starting in $\mathbf{c}$ either in forward or in backward direction ([8, 13]). For intersecting stream surfaces, the problem states as follows: given $\mathbf{w}$ and two stream surfaces defined by the seeding curves $\mathbf{c}_1$ (in forward direction) and $\mathbf{c}_2$ (in backward direction), their intersection is in general a finite number of stream lines[1] ([18]). Each of them starts at $\mathbf{c}_1$ and ends (by applying a forward integration) in $\mathbf{c}_2$. Figure 2 gives an illustration.



(a) Seeding lines $\mathbf{c}_1$ and $\mathbf{c}_2$.

(b) Stream surfaces starting from $\mathbf{c}_1$ (forwards) and $\mathbf{c}_2$ (backwards).

(c) Intersection curve is a stream line.

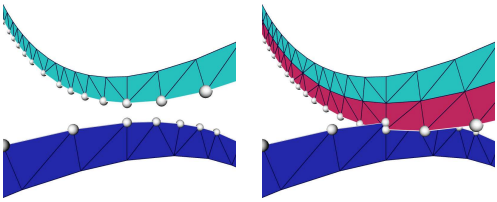
Figure 2: Intersecting stream surfaces in a 3D vector field.

[18] describes a numerical stream surface intersection approach in the context of particular stream surfaces – namely separation surfaces starting at 3D saddle points. Fortunately, this algorithm can directly be extended to extract intersections of general stream surfaces. So we describe the main idea here and refer to [18] for details.

To find the intersection between a stream surfaces in forward integration and a stream surface in backward integration, we integrate both stream surfaces simultaneously until an intersection point $\mathbf{p}_1$ is found. After refining this point (see [18]), a stream line from $\mathbf{p}_1$ is integrated both backward and forward until it reaches $\mathbf{c}_1$ and $\mathbf{c}_2$ respectively. The algorithm stops if one of the stream surfaces completely leaves the domain of $\mathbf{w}$, if a (user defined) maximal number of intersection curves is found, or if a (user-defined) maximal number of numerical stream surface integration steps have been performed.

To find the intersection of stream surfaces, we only have to consider the evolving front of the currently integrated stream surface. This front is represented as a triangular strip. After each integration step, the front strip is checked for intersections with the front strip of the other stream surface. Figure 3 illustrates this.

[1] Here we do not consider the case of partially collapsing stream surfaces.



(a) Front ribbons shortly before an intersection is found.

(b) Intersection is found.

Figure 3: Intersecting stream surfaces (from [18]).

3 The Algorithm

In this section we describe the main ingredients of our approach. Given a 2D vector field

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

and a line

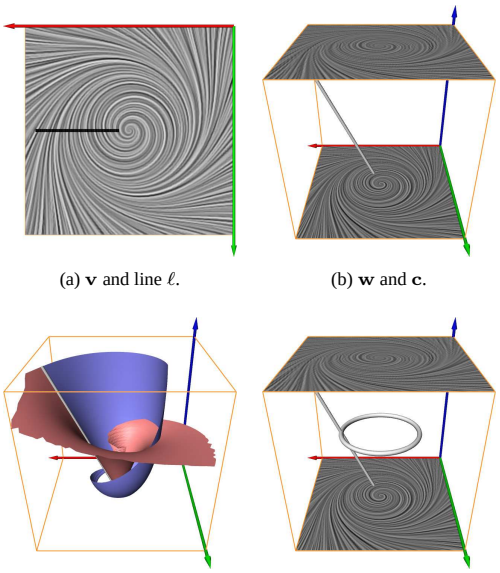
$$\ell(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

in the domain of $\mathbf{v}$, we want to find all closed stream lines in $\mathbf{v}$ which cross ℓ. To do so, we transform $\mathbf{v}$ into a 3D vector field $\mathbf{w}$, and ℓ into a 3D seeding curve $\mathbf{c}$ in the following way:

$$\mathbf{w}(x,y,z) = \begin{pmatrix} u(x,y) \\ v(x,y) \\ 0 \end{pmatrix}, \quad \mathbf{c}(t) = \begin{pmatrix} x(t) \\ y(t) \\ t \end{pmatrix}.$$

Note that $\mathbf{c}$ is strictly monotonous in z-direction: for each z-value there is only one point on $\mathbf{c}$. Then we apply the stream surface intersection described in section 2 in $\mathbf{w}$ with the seeding lines $\mathbf{c}_1 = \mathbf{c}_2 = \mathbf{c}(t)$. This way, each detected intersection curve in $\mathbf{w}$ corresponds to a closed stream line in $\mathbf{v}$. Since a line in $\mathbf{w}$ always consists of points with the same z-value, simply an omitting of the z-values of the found intersection curves in $\mathbf{w}$ yields the closed stream lines in $\mathbf{v}$. Figure 4 illustrates this algorithm.

Note that the algorithm detects the same closed stream line more than one time if it crosses ℓ more than one time. Figure 5 shows an example. The removal of multiple detected closed stream lines in $\mathbf{w}$ can be done by projecting them onto the plane $z = 0$ and comparing their Hausdorff distance: if it is under a certain small threshold, one of the closed curves is deleted.



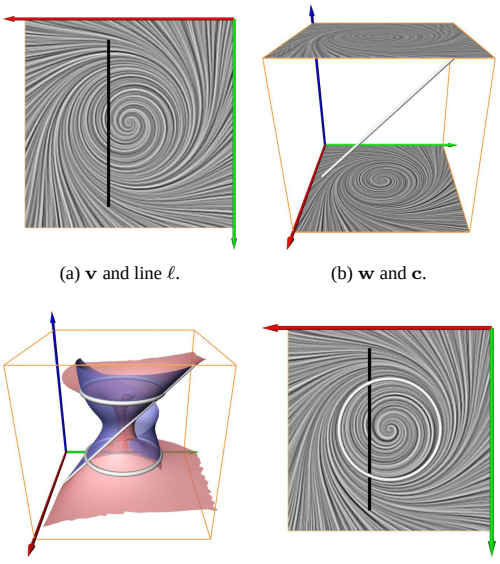
(a) $\mathbf{v}$ and line ℓ.

(b) $\mathbf{w}$ and $\mathbf{c}$.

(c) Stream surfaces started from $\mathbf{c}$ in both forward and backward direction.

(d) Detected intersection curve is closed stream line.

Figure 4: Detecting closed stream lines crossing a line ℓ.



(a) $\mathbf{v}$ and line ℓ.

(b) $\mathbf{w}$ and $\mathbf{c}$.

(c) Intersecting stream surfaces and closed stream line detected two times.

(d) Projection onto $z = 0$. One closed stream line remains.

Figure 5: Detection and projection of a closed stream line crossing a line ℓ twice.

4 Choosing the Seeding Lines

Up to now we are able to detect all closed stream lines which cross a certain given line ℓ. What remains is to choose ℓ in such a way that *all* closed stream lines in $\mathbf{v}$ are guaranteed to be detected. To do so, we use the following

Theorem 1 *Given a 2D vector field* $\mathbf{v}$*, inside each isolated closed stream line there must be at least one critical point with index* $+1$*, i.e. a source, sink or center.*

This theorem follows directly from the index theorem of 2D vector fields [1]: considering the area of $\mathbf{v}$ inside a closed stream line, this area has a global index of $+1$. Hence, at least one critical point with the index $+1$ must be contained within the area inside the closed stream line.

Because of theorem 1, a line ℓ with the following conditions detects all closed stream lines of $\mathbf{v}$:

- All critical points of $\mathbf{v}$ with an index $+1$ are on ℓ.
- Either the start or end point of ℓ is on the boundary of the domain of $\mathbf{v}$.

Here we construct ℓ as a polygon in the following way:

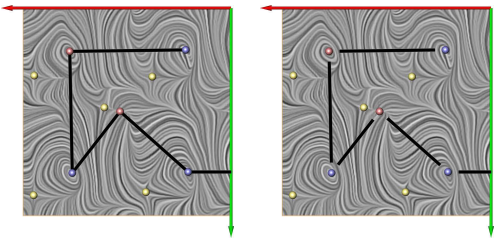
1. Detect all critical points $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n$ of $\mathbf{v}$ with an index $+1$.
2. Detect the point $\mathbf{p}_0$ on the boundary of the domain of $\mathbf{v}$ which is closest to $\mathbf{p}_1$.
3. The polygon $(\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_n)$ is the line ℓ (Figure 6a).

Note that ℓ may have self-intersections and a rather strange shape, due to the fact that the ordering of the detected critical points is arbitrary. However, the 3D seeding curve $\mathbf{c}$ derived from ℓ (see section 3) does not have self-intersections.

As the integration at and near the critical points is numerically unstable, these small areas must be excluded (Figures 6a-b). To achieve this, we split the polygon ℓ into its line segments $(\mathbf{p}_i, \mathbf{p}_{i+1})$. Every $\mathbf{p}$ coinciding with a critical point is moved away from it by a small amount ε along the line segment vector $\mathbf{s}_i = \mathbf{p}_{i+1} - \mathbf{p}_i$. This is done for every segment separately. Thus, we yield the new segments:

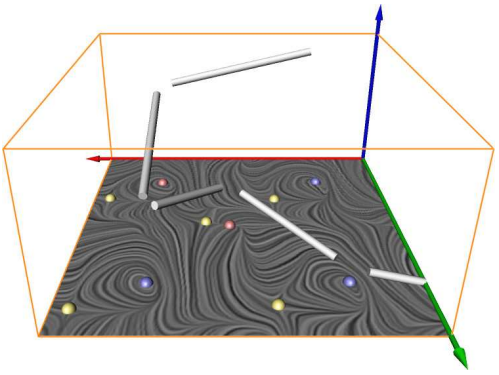
$$\left(\mathbf{p}_i + \varepsilon \cdot \frac{\mathbf{s}_i}{|\mathbf{s}_i|} \;,\; \mathbf{p}_{i+1} - \varepsilon \cdot \frac{\mathbf{s}_i}{|\mathbf{s}_i|} \right).$$

There are two ways to transform the line segments into 3D seeding curves:

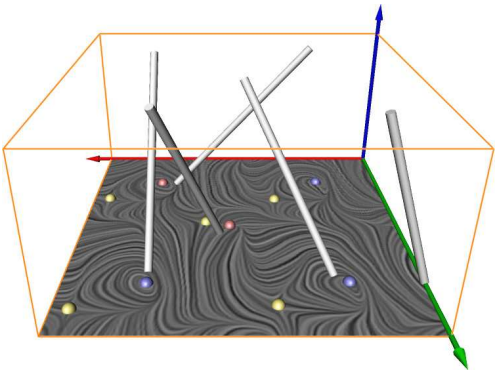


(a) Polygon ℓ connecting all sources/sinks and one boundary point.

(b) Polygon ℓ splitted around the critical points.



(c) Seeding curves: Transformation according to location on ℓ.



(d) Seeding curves: Transformation according to location on line segment.

Figure 6: Finding seeding lines to detect all closed stream lines of $\mathbf{v}$.

1. Transform each point according to its location t on ℓ. This yields seeding curves with non-overlapping z-value ranges. Thus, the algorithm of section 3 can be applied in a parallel manner. (Figure 6c)

2. Transform each point according to its location t on the line segment itself. The resulting seeding curves have overlapping z-value ranges. Therefore, the algorithm needs to be applied in a serial manner, i.e. each curve must be treated separately. (Figure 6d)

We found the latter type of transformation to be more robust, as the seedings curves have a fixed z-value range and a guaranteed minimal length. In this case, choosing a fixed resolution for the stream surface integration of all curves yields to robust results for our test data sets. Though a different resolution for each curve depending on the line segment length might speed up the computation in some areas.

5 Applications

Figures 6 and 7 visualize a random 2D data set defined on a 20×16 grid. Random vector fields are useful tools for a proof-of-concept of topological methods, since they contain a maximal amount of topological information.

We detected 5 saddle points, 3 sinks and 2 sources in this data set. They can be distinguished in the color plot of figure 7c: saddles are yellow, sinks are blue and sources are red. Sources and sinks have an index of $+1$. How the polygon ℓ and the corresponding seeding curves are constructed from this information can be seen in figure 6. Note, that here the excluded area around the critical points has been enlarged for demonstration purposes only. For the calculations presented in figure 7 the excluded area is an order of magnitude smaller. This difference can be seen by comparing figures 6d and 7d.

Figures 7a-b show the stream surfaces emanating from the seeding curves. As discussed in section 4 only the two surfaces coming from the same seeding curve are tested for intersection against each other.

Figure 7d shows in conjunction with figure 7c that our algorithm found a stream surface intersection at all places where a seeding curve crosses a closed stream line. Thus, some of the closed stream lines are detected multiple times. As explained in section 3 this can be dealt with by projecting them onto the plane $z = 0$ and comparing their Hausdorff distance. This reduces the number of 9 found intersections to 5 unique closed stream lines – as it can be seen in figure 7c. Our algorithm needed approximately 90 seconds to extract the closed stream lines (Pentium 4, 1.7GHz).

One of the closed stream lines in the upper left corner of figure 7c is very close to a critical point (a source). Figure 7e shows a closeup of that area. This exemplifies that our algorithm works reliable even in a rather small distance away from a critical point.

6 Conclusions

We have introduced a new approach to detecting closed stream lines in 2D vector fields. Based on the fact that this detection is equivalent to intersecting certain stream surfaces in appropriate 3D vector fields, we were able to apply a previously developed numerical stream surface intersection algorithm for our purposes. The resulting approach does not depend on any underlying grid structure of the vector field.

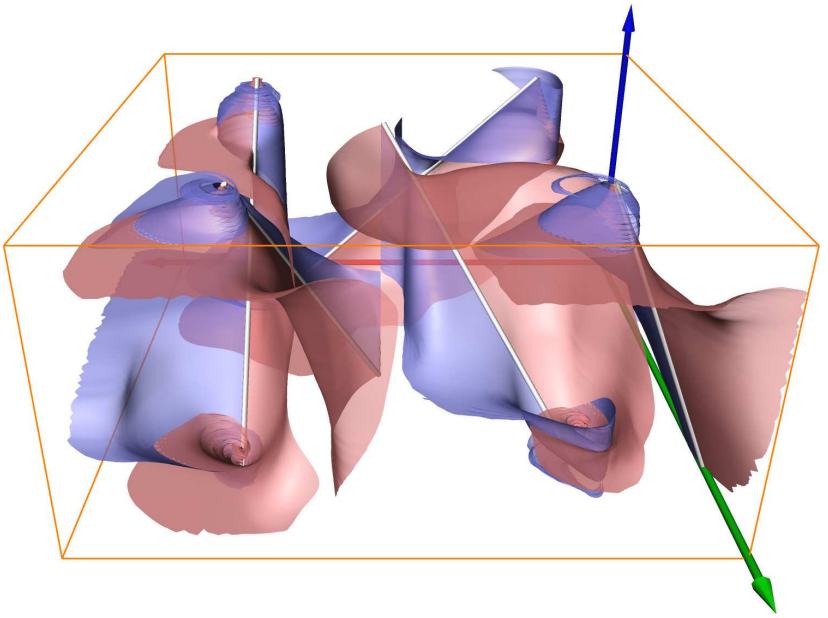
In particular, our approach fills a gap in the closed stream line tracking approach for 2D time-dependent vector fields described in [19]. There the tracking started at Hopf and periodic blue sky bifurcations. Since closed stream lines may already be present in the first time step, a stable detection approach for a particular time step is necessary to guarantee to track all closed stream lines.

Future research clearly goes into the direction of detecting closed structures in 3D vector fields. The extension of our approach to 3D vector fields seems to be possible but is not straightforward, since in this case 3D stream-hypersurfaces of 4D vector fields have to be intersected.

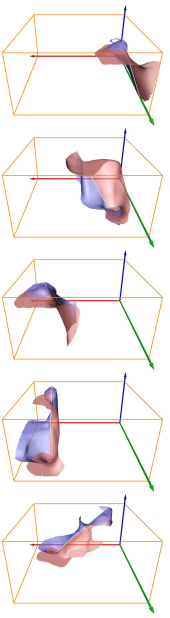
Acknowledgments

We thank Bernd R. Noack, Ivanka Pelivan and Jan Sahner for the fruitful discussions.

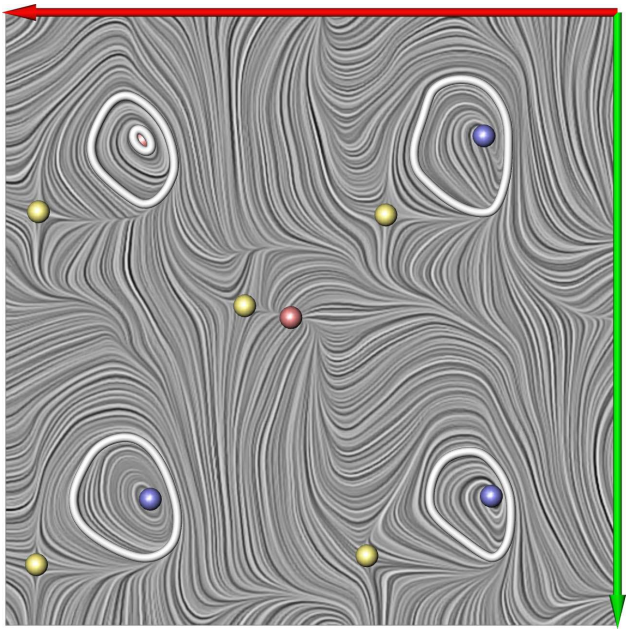
All visualizations in this paper have been created using AMIRA – a system for advanced 3D visualization and volume modeling [14] (see `http://amira.zib.de/`).



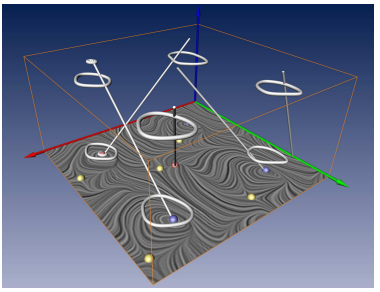
(a) Intersecting stream surfaces of all 5 seeding curves shown at once. They must be treated separately as they have overlapping z-value ranges.



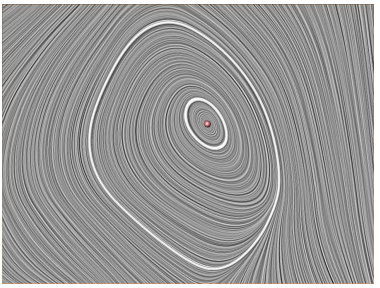
(b) Each curve treated separately.



(c) Projection onto $z = 0$: 5 closed stream lines remain.



(d) 9 detected intersection curves.



(e) Closeup of the upper left corner.

Figure 7: Test data set: 5 closed stream lines have been detected.

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. Batra, K. Kling, and L. Hesselink. Topology based vector field comparison using graph methods. In *Proc. IEEE Visualization '99, Late Breaking Hot Topics*, pages 25–28, 1999.

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

[4] W. de Leeuw and R. van Liere. Visualization of global flow structures using multiple levels of topology. In *Data Visualization 1999. Proc. VisSym 99*, pages 45–52, 1999.

[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. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11:36–46, May 1991.

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

[9] Y. Lavin, R.K. Batra, and L. Hesselink. Feature comparisons of vector fields using earth mover's distance. In *Proc. IEEE Visualization '98*, pages 103–109, 1998.

[10] 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.

[11] K. Mahrous, J. Bennett, B. Hamann, and K. Joy. Improving topological segmentation of three-dimensional vector fields. In *Data Visualization 2003. Proc. VisSym 03*, pages 203–212, 2003.

[12] 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.

[13] D. Stalling. *Fast Texture-based Algorithms for Vector Field Visualization*. PhD thesis, FU Berlin, Department of Mathematics and Computer Science, 1998.

[14] D. Stalling, H.-C. Hege, and M. Westerhoff. Amira – a highly interactive system for visual data analysis. In Christopher R. Johnson and Charles D. Hansen, editors, *Visualization Handbook*. Academic Press, 2004.

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

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

[17] H. Theisel and T. Weinkauf. Vector field metrics based on distance measures of first order critical points. In *Journal of WSCG*, volume 10:3, pages 121–128, 2002.

[18] 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.

[19] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Stream line and path line oriented topology for 2D time-dependent vector fields. In *Proc. IEEE Visualization 2004*, 2004.

[20] X. Tricoche, G. Scheuermann, and H. Hagen. A topology simplification method for 2D vector fields. In *Proc. IEEE Visualization 2000*, pages 359–366, 2000.

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

[22] X. Tricoche, G. Scheuermann, and H.Hagen. Topology-based visualization of time-dependent 2D vector fields. In *Data Visualization 2001. Proc. VisSym 01*, pages 117–126, 2001.

[23] 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.

[24] 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*, 2004.

[25] 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), 2004.

[26] 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.

[27] 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.

[28] T. Wischgoll, G. Scheuermann, and H. Hagen. Tracking closed stream lines in time-dependent planar flows. In *Proc. Vision, Modeling and Visualization 2001*, pages 447–454, 2001.