Ronald Peikert   Hamish Carr
Helwig Hauser   Raphael Fuchs   *Editors*

Topological Methods in Data Analysis and Visualization II

Theory, Algorithms, and Applications

Springer

Mathematics and Visualization

Series Editors

Gerald Farin
Hans-Christian Hege
David Hoffman
Christopher R. Johnson
Konrad Polthier
Martin Rumpf

For further volumes:
http://www.springer.com/series/4562

Ronald Peikert
Helwig Hauser
Hamish Carr
Raphael Fuchs

Editors

Topological Methods in Data Analysis and Visualization II

Theory, Algorithms, and Applications

Springer

Editors

Ronald Peikert
ETH Zürich
Computational Science
Zürich
Switzerland
peikert@inf.ethz.ch

Hamish Carr
University of Leeds
School of Computing
Leeds
United Kingdom
H.Carr@leeds.ac.uk

Helwig Hauser
University of Bergen
Dept. of Informatics
Bergen
Norway
Helwig.Hauser@UiB.no

Raphael Fuchs
ETH Zürich
Computational Science
Zürich
Switzerland
raphael@inf.ethz.ch

Preface

Over the past few decades, scientific research became increasingly dependent on large-scale numerical simulations to assist the analysis and comprehension of physical phenomena. This in turn has led to an increasing dependence on scientific visualization, i.e., computational methods for converting masses of numerical data to meaningful images for human interpretation.

In recent years, the size of these data sets has increased to scales which vastly exceed the ability of the human visual system to absorb information, and the phenomena being studied have become increasingly complex. As a result, scientific visualization, and scientific simulation which it assists, have given rise to systematic approaches to recognizing physical and mathematical features in the data.

Of these systematic approaches, one of the most effective has been the use of a topological analysis, in particular computational topology, i.e., the topological analysis of discretely sampled and combinatorially represented data sets. As topological analysis has become more important in scientific visualization, a need for specialized venues for reporting and discussing related research has emerged.

This book results from one such venue: the *Fourth Workshop on Topology Based Methods in Data Analysis and Visualization (TopoInVis 2011)*, which took place in Zürich, Switzerland, on April 4–6, 2011. Originating in Europe with successful workshops in Budmerice, Slovakia (2005), and Grimma, Germany (2007), this workshop became truly international with TopoInVis 2009 in Snowbird, Utah, USA (2009). With 43 participants, TopoInVis 2011 continues this run of successful workshops, and future workshops are planned in both Europe and North America under the auspices of an international steering committee of experts in topological visualization, and a dedicated website at http://www.TopoInVis.org/.

The program of *TopoInVis 2011* included 20 peer-reviewed presentations and two keynote talks given by invited speakers. Martin Rasmussen, Imperial College, London, addressed the ongoing efforts of our community to formulate a vector field topology for unsteady flow. His presentation *An introduction to the qualitative theory of nonautonomous dynamical systems* was highly appreciated as an illustrative introduction into a difficult mathematical subject. The second keynote, *Looking for intuition behind discrete topologies*, given by Thomas Lewiner, PUC-Rio,

Rio de Janeiro, picked up another topic within the focus of current research, namely combinatorial methods, for which his talk gave strong motivation. At the end of the workshop, Dominic Schneider and his coauthors were given the award for the best paper by a jury.

Nineteen of the papers presented at *TopoInVis 2011* were revised and, in a second round of reviewing, accepted for publication in this book. Based on the major topics covered, the papers have been grouped into four parts.

The first part of the book is concerned with computational discrete Morse theory, both in 2D and in 3D. In 2D, Reininghaus and Hotz applied discrete Morse theory to divergence-free vector fields. In contrast, Günther et al. present a combinatorial algorithm to construct a hierarchy of combinatorial gradient vector fields in 3D, while Gyulassy and Pascucci provide an algorithm that computes the distinct cells of the MS complex connecting two critical points. Finally, an interesting contribution is also made by Reich et al. who developed a combinatorial vector field topology in 3D.

In Part 2, hierarchical methods for extracting and visualizing topological structures such as the contour tree and Morse-Smale complex were presented. Weber et al. propose an enhanced method for contour trees that is able to visualize two additional scalar attributes. Harvey et al. introduce a new clustering-based approach to approximate the Morse–Smale complex. Finally, Wagner et al. describe how to efficiently compute persistent homology of cubical data in arbitrary dimensions.

The third part of the book deals with the visualization of dynamical systems, vector and tensor fields. Tricoche et al. visualize chaotic structures in area-preserving maps. The same problem was studied by Sanderson et al. in the context of an application, namely the structure of magnetic field lines in tokamaks, with a focus on the detection of islands of stability. Jadhav et al. present a complete analysis of the possible mappings from inflow boundaries to outflow boundaries in triangular cells. A novel algorithm for pathline placement with controlled intersections is described by Weinkauf et al., while Wiebel et al. propose glyphs for the visualization of nonlinear vector field singularities. As an interesting result in tensor field topology, Lin et al. present an extension to asymmetric 2D tensor fields.

The final part is dedicated to the topological visualization of unsteady flow. Kasten et al. analyze finite-time Lyapunov exponents (FTLE) and propose alternative realizations of Lagrangian coherent structures (LCS). Schindler et al. investigate the flux through FTLE ridges and propose an efficient, high-quality alternative to height ridges. Pobitzer et al. present a technique for detecting and removing false positives in LCS computation. Schneider et al. propose an FTLE-like method capable of handling uncertain velocity data. Sadlo et al. investigate the time parameter in the FTLE definition and provide a lower bound. Finally, Fuchs et al. explore scale-space approaches to FTLE and FTLE ridge computation.

team provided valuable support by setting up the registration web page and promptly resolving issues with on-line payments. We are grateful to Marianna Berger, Katharina Schuppli, Robert Carnecky, and Benjamin Schindler for their administrative and organizational help. We also wish to thank the *TopoInVis* steering committee for their advice and their help with advertising the event. The project *SemSeg–4D Space-Time Topology for Semantic Flow Segmentation* supported TopoInVis 2011 in several ways, most notably by offering 12 young researchers partial refunding of their travel costs. The project SemSeg acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 226042.

We are looking forward to the next *TopoInVis* workshop, which is planned to take place in 2013 in North America.

Ronald Peikert
Helwig Hauser
Hamish Carr
Raphael Fuchs

Contents

ix

Part I
Discrete Morse Theory

Computational Discrete Morse Theory
for Divergence-Free 2D Vector Fields

Jan Reininghaus and Ingrid Hotz

1 Introduction

We introduce a robust and provably consistent algorithm for the topological analysis of divergence-free 2D vector fields.

Topological analysis of vector fields has been introduced to the visualization community in [10]. For an overview of recent work in this field we refer to Sect. 2. Most of the proposed algorithms for the extraction of the topological skeleton try to find all zeros of the vector field numerically and then classify them by an eigenanalysis of the Jacobian at the respective points. This algorithmic approach has many nice properties like performance and familiarity. Depending on the data and the applications there are however also two shortcomings.

1.1 Challenges

If the vector field contains plateau like regions, i.e. regions where the magnitude is rather small, these methods have to deal with numerical problems and may lead to topologically inconsistent results. This means that topological skeletons may be computed that cannot exist on the given domain. A simple example for this problem can be given in 1D. Consider an interval containing exactly three critical points as shown in Fig. 1a. While it is immediately clear that not all critical points can be of the same type, an algorithm that works strictly locally using numerical algorithms may result in such an inconsistent result. A second problem that often arises is that

J. Reininghaus (✉) · I. Hotz
Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
e-mail: reininghaus@zib.de; hotz@zib.de
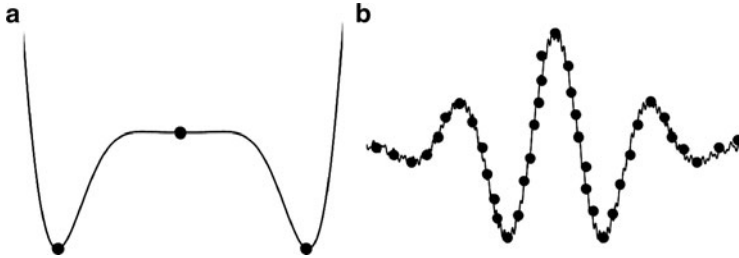
Fig. 1 Illustration of the algorithmic challenges. (**a**) shows 1D function with a plateau-like region. From the topological point of view the critical point in the middle needs to be a maximum since it is located between the two minima on the left and on the right side. However, depending on the numerical procedure the determination of its type might be inconsistent. (**b**) illustrates a noisy 1D function. Every fluctuation caused by the noise generates additional minima and maxima

of noise in the data. Depending on its type and quantity, a lot of spurious critical points may be produced as shown in Fig. 1b. Due to the significance of this problem in practice, a lot of work has been done towards robust methods that can deal with such data, see Sect. 2.

1.2   Contribution

This paper proposes an application of computational discrete Morse theory for divergence-free vector fields. The resulting algorithm for the topological analysis of such vector fields has three nice properties:

1. It provably results in a set of critical points that is consistent with the topology of the domain. This means that the algorithm cannot produce results that are inadmissible on the given domain. The consistency of the algorithm greatly increases its robustness as it can be interpreted as an error correcting code. We will give a precise definition of topological consistency for divergence-free vector fields in Sect. 3.
2. It allows for a simplification of the set of critical points based on an importance measure related to the concept of persistence [5]. Our method may therefore be used to extract the structurally important critical points of a divergence-free vector field and lends itself to the analysis of noisy data sets. The importance measure has a natural physical interpretation and is described in detail in Sect. 4.
3. It is directly applicable to vector fields with only near zero divergence. These fields often arise when divergence-free fields are numerically approximated or measured. This property in demonstrated in Sect. 5.

2   Related Work

Vector field topology was introduced to the visualization community by Helman and Hesselink [11]. They defined the concept of a topological skeleton consisting of critical points and connecting separatrices to segment the field into regions of topologically equivalent streamline behavior. A good introduction to the concepts and algorithms of vector field topology is given in [30], while a systematic survey of recent work in this field can be found in [15].

As the topological skeleton of real world data sets is usually rather complex, a lot of work has been done towards simplification of topological skeletons of vector fields, see [14, 28, 29, 31].

To reduce the dependence of the algorithms on computational parameters like step sizes, a combinatorial approach to vector field topology based on Conley index theory has been developed [3, 4]. In the case of divergence-free vector fields their algorithm unfortunately encounters many problems in practice.

For scalar valued data, algorithms have been developed [1, 8, 13, 16, 25] using concepts from discrete Morse theory [7] and persistent homology [5]. The basic ideas in these algorithms have been generalized to vector valued data in [23, 24] based on a discrete Morse theory for general vector fields [6]. This theory however is not applicable for divergence-free vector fields since it does not allow for center-like critical points. Recently, a unified framework for the analysis of vector fields and gradient vector fields has been proposed in [22] under the name computational discrete Morse theory.

Since vector field data is in general defined in a discrete fashion, a discrete treatment of the differential concepts that are necessary in vector field topology has been shown to be beneficial in [21, 27]. They introduced the idea that the critical points of a divergence-free vector field coincide with the extrema of the scalar potential of the point-wise-perpendicular field to the visualization community. The critical points can therefore be extracted by reconstructing this scalar potential and extracting its minima, maxima, and saddle points. In contrast to our algorithm, their approach does not exhibit the three properties mentioned in Sect. 1.

3   Morse Theory for Divergence-Free 2D Vector Fields

This section shows how theorems from classical Morse theory can be applied in the context of 2D divergence-free vector fields.

3.1   Vector Field Topology

A 2D vector field v is called divergence-free if $\nabla \cdot v = 0$. This class of vector fields often arises in practice, especially in the context of computational fluid dynamics.

For example, the vector field describing the flow of an incompressible fluid, like water, is in general divergence-free. The points at which a vector field v is zero are called the critical points of v. They can be classified by an eigenanalysis of the Jacobian Dv at the respective critical point. In the case of divergence-free 2D vector fields one usually distinguishes two cases [10]. If both eigenvalues are real, then the critical point is called a saddle. If both eigenvalues are imaginary, then the critical point is called a center. Note that one can classify a center furthermore into clockwise rotating (CW-center) or counter-clockwise rotating (CCW-center) by considering the Jacobian as a rotation.

One consequence of the theory that will be presented in this section is that the classification of centers into CW-centers and CCW-centers is essential from a topological point of view. One can even argue that this distinction is as important as differentiating between minima and maxima when dealing with gradient vector fields.

3.2 Morse Theory

The critical points of a vector field are often called topological features. One justification for this point of view is given by Morse theory [17]. Loosely speaking, Morse theory relates the set of critical points of a vector field to the topology of the domain. For example, it can be proven that every continuous vector field on a sphere contains at least one critical point. To make things more precise we restrict ourselves to gradient vector fields defined on a closed oriented surface. The ideas presented below work in principal also for surfaces with boundary, but the notation becomes more cumbersome. To keep things simple, we therefore assume that the surface is closed. We further assume that all critical points are first order, i.e. the Jacobian has full rank at each critical point. Let c_0 denote the number of minima, c_1 the number of saddles, c_2 the number of maxima, and g the genus of the surface. We then have the Poincaré-Hopf theorem

$$c_2 - c_1 + c_0 = 2 - 2g, \tag{1}$$

the weak Morse inequalities

$$c_0 \geq 1, \qquad c_1 \geq 2g, \qquad c_2 \geq 1, \tag{2}$$

and the strong Morse inequality

$$c_1 - c_0 \geq 2g - 1. \tag{3}$$

3.3 Helmholtz-Hodge Decomposition

To apply these theorems from Morse theory to a divergence-free vector field v we can make use of the Helmholtz-Hodge decomposition [12]. Let $\nabla \times \psi = (\partial_y \psi, -\partial_x \psi)$ denote the curl operator in 2D. We then have the orthogonal decomposition

$$v = \nabla \phi + \nabla \times \psi + h. \tag{4}$$

We can thereby uniquely decompose v into an irrotational part $\nabla \phi$, a solenoidal part $\nabla \times \psi$, and a harmonic part h, i.e. $\Delta h = 0$. Due to the assumption that the surface is closed, the space of harmonic vector fields coincides with the space of vector fields with zero divergence and zero curl [26]. Since v is assumed to be divergence-free we have $0 = \nabla \cdot v = \nabla \cdot \nabla \phi$ which implies $\phi = 0$ due to (4). The harmonic-free part $\hat{v} = v - h$ can therefore be expressed as the curl of a scalar valued function

$$\hat{v} = \nabla \times \psi. \tag{5}$$

3.4 Stream Function

The function ψ is usually referred to as the stream function [19]. Let $\hat{v}^\perp = (v_2, -v_1)$ denote the point-wise perpendicular vector field of $\hat{v} = (v_1, v_2)$. The gradient of the stream function is then given by

$$\nabla \psi = \hat{v}^\perp. \tag{6}$$

Note that $\hat{v}$ has the same set of critical points as $\hat{v}^\perp$. The type of its critical points is however changed: CW-center become minima, and CCW-center become maxima. Since (6) shows that $\hat{v}^\perp$ is a gradient vector field, we can use this identification to see how (1)–(3) can be applied to the harmonic-free part of divergence-free 2D vector fields.

3.5 Implications

The dimension of the space of harmonic vector fields is given by $2g$ [26]. A vector field defined on a surface which is homeomorphic to a sphere is therefore always harmonic-free, i.e. $\hat{v} = v$. Every divergence-free vector field on a sphere which only contains first order critical points therefore satisfies (1)–(3). For example, every such vector field contains at least one CW-center and one CCW-center.

Due to the practical relevance in Sect. 5 we note that every divergence-free vector field defined on a contractible surface can be written as the curl of a stream function ψ as shown by the Poincaré-Lemma. For such cases, the point-wise perpendicular vector field can therefore also be directly interpreted as the gradient of the stream function.

4 Algorithmic Approach

4.1 Overview

We now describe how we can apply computational discrete Morse theory to divergence-free vector fields. Let v denote a divergence-free vector field defined on an oriented surface S. The first step is to compute the harmonic-free part $\hat{v}$ of v. If S is contractible or homeomorphic to a sphere, then v is itself the curl of a stream function ψ, i.e. $\hat{v} = v$. Otherwise, we need to compute the Helmholtz-Hodge decomposition (4) of v to get its harmonic part. To do this, one can employ the algorithms described in [20, 21, 27].

We now make use of the fact that the point-wise perpendicular vector field $\hat{v}^{\perp}$ has the same critical points as $\hat{v}$. Due to (5), we know that $\hat{v}^{\perp}$ is a gradient vector field. To compute and classify the critical points of the divergence-free vector field $\hat{v}$ it therefore suffices to analyze the gradient vector field $\hat{v}^{\perp}$.

One approach to analyze the gradient vector field $\hat{v}^{\perp}$ would be to compute a scalar valued function ψ such that $\hat{v}^{\perp} = \nabla\psi$. One can then apply one of the algorithms mentioned in Sect. 2 to extract a consistent set of critical points. In this paper, we will apply an algorithm from computational discrete Mose theory to directly analyze the gradient vector field $\hat{v}^{\perp}$. The main benefit of this approach is that it allows us to consider $\hat{v}^{\perp}$ as a gradient vector field even if it contains a small amount of curl. This is a common problem in practice, since a numerical approximation or measurement of a divergence-free field often contains a small amount of divergence. By adapting the general approach presented in [22], we can directly deal with such fields with no extra pre-processing steps. Note that the importance measure for the critical points of a gradient vector field has a nice physical interpretation in the case of rotated stream functions. This will be explained in more detail below.

4.2 Computational Discrete Morse Theory

The basic idea in computational discrete Morse theory is to consider Forman's discrete Morse theory [7] as a discretization of the admissible extremal structures of a given surface. The extremal structure of a scalar field consists of critical points and separatrices – the integral lines of the gradient field that connect the critical points. Using this description of the topologically consistent structures we then define an optimization problem that results in a hierarchy of extremal structures that represents the given input data with decreasing level of detail.

4.2.1 Definitions

Let C denote a finite regular cell complex [9] that represents the domain of the given vector field. Examples of such cell complexes that arise in practice are triangulations or quadrangular meshes. We first define its cell graph $G = (N, E)$, which encodes the combinatorial information contained in C in a graph theoretic setting.

The nodes N of the graph consist of the cells of the complex C and each node u^p is labeled with the dimension p of the cell it represents. The edges E of the graph encode the neighborhood relation of the cells in C. If the cell u^p is in the boundary of the cell w^{p+1}, then $e^p = \{u^p, w^{p+1}\} \in E$. We refer to Fig. 2a for an example of a simple cell graph. Note that we additionally label each edge with the dimension of its lower dimensional node.

A subset of pairwise non-adjacent edges is called a matching. Using these definitions, a combinatorial vector field V on a regular cell complex C can be defined as a matching of the cell graph G, see Fig. 2a for an example. The set of combinatorial vector fields on C is thereby given by the set of matchings $\mathcal{M}$ of the cell graph G.

We now define the extremal structure of a combinatorial vector field. The unmatched nodes are called critical points. If u^p is a critical point, we say that the critical point has index p. A critical point of index p is called sink ($p = 0$), saddle ($p = 1$), or source ($p = 2$). A combinatorial p-streamline is a path in the graph whose edges are of dimension p and alternate between V and its complement. A p-streamline connecting two critical points is called a p-separatrix. If a p-streamline is closed, we call it either an attracting periodic orbit ($p = 0$) or a repelling periodic orbit ($p = 1$). For examples of these combinatorial definitions of the extremal structure we refer to Figs. 2b–d.

As shown in [2], a combinatorial gradient vector field V^ϕ can be defined as a combinatorial vector field that contains no periodic orbits. A matching of G that gives rise to such a combinatorial vector field is called a Morse matching. The set of combinatorial gradient vector fields on C is therefore given by the set of Morse matchings $\mathcal{M}$ of the cell graph G. In the context of gradient vector fields, we refer to a critical point u^p as a minimum ($p = 0$), saddle ($p = 1$), or maximum ($p = 2$).
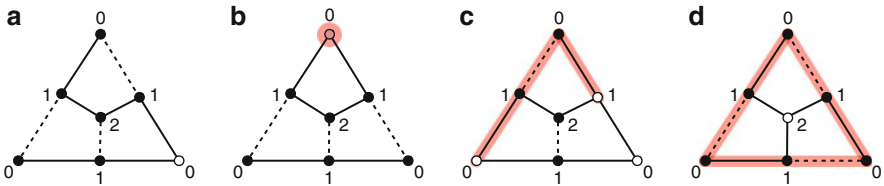
Fig. 2 Basic definitions. (**a**) a combinatorial vector field (*dashed*) on the cell graph of a single triangle. The numbers correspond to the dimension of the represented cells, and matched nodes are drawn solid. (**b**) a critical point of index 0. (**c**) a 0-separatrix. (**d**) an attracting periodic orbit

We now compute edge weights $\omega : E \to \mathbb{R}$ to represent the given vector field $\hat{v}^\perp$. The idea is to assign a large weight to an edge $e^p = \{u^p, w^{p+1}\}$ if an arrow pointing from u^p to w^{p+1} represents the flow of $\hat{v}^\perp$ well. The weight for e^p is therefore computed by integrating the tangential component of the vector field $\hat{v}^\perp$ along the edge e^p.

4.2.2 Computation

We can now define the optimization problem

$$V_k^\phi = \arg \max_{M \in \mathscr{M}^\phi, |M|=k} \omega(M). \tag{7}$$

Let $k_0 = \arg \max_{k \in \mathbb{N}} \omega(V_k)$ denote the size of the maximum weight matching, and let $k_n = \max_{k \in \mathbb{N}} |V_k|$ denote the size of the heaviest maximum cardinality matching. The hierarchy of combinatorial gradient vector fields that represents the given vector field $\hat{v}^\perp$ with decreasing level of detail is now given by

$$\mathscr{V}^\phi = \left(V_k^\phi \right)_{k=k_0,\ldots,k_n}. \tag{8}$$

For a fast approximation algorithm for (8) and the extraction of the extremal structure of a particular combinatorial gradient vector field we refer to [22].

4.2.3 Importance Measure

Note that the sequence (8) is ordered by an importance measure which is closely related to homological persistence [5]. The importance measure is defined by the height difference of a certain pairing of critical points. Since we are dealing with the gradient of a stream function of a divergence-free vector field there is a nice physical interpretation of this value. The height difference between two points of the stream function is the same as the amount of flow passing through any line connecting the two points [19]. This allows us to differentiate between spurious and structurally important critical points in divergence-free 2D vector fields, as will be demonstrated in the next section.

5 Examples

The purpose of this section is to provide some numerical evidence for the properties of our method mentioned in Sect. 1. The running time of our algorithm is 47 s for a surface with one million vertices using an Intel Core i7 860 CPU with 8 GB RAM.

5.1 Noise Robustness

To illustrate the robustness of our algorithm with respect to noise, we sampled the divergence-free vector field

$$v(x, y) = \nabla \times \left(\sin(6\,x)\, \sin(6\,y)\, e^{-3\,(x^2 + y^2)} \right) \qquad (9)$$

on the domain $[-1, 1]^2$ with a uniform 512^2 grid. A LIC image of this divergence-free vector field is shown in Fig. 3, left. To simulate a noisy measurement of this vector field, we added uniform noise with a range of $[-1, 1]$ to this data set. A LIC image of the resulting quasi-divergence-free vector field is shown in Fig. 3, right. Since the square is a contractible domain, we can directly apply the algorithm described in Sect. 4 to both data sets and extracted the 23 most important critical points. As can be seen in Fig. 3, our method is able to effectively deal with the noisy data.

5.2 Importance Measure

To illustrate the physical relevance of the importance measure for the extracted critical points we consider a model example from computational fluid dynamics [18]. Figure 4, top, shows a LIC image of a simulation of the flow behind a circular cylinder – the cylinder is on the left of the shown data set. Since we are considering only a contractible subset of the data set, we can directly apply the algorithm described in Sect. 4. Note that due to a uniform sampling of this data set a small amount of divergence was introduced. The divergence is depicted in Fig. 4, bottom.
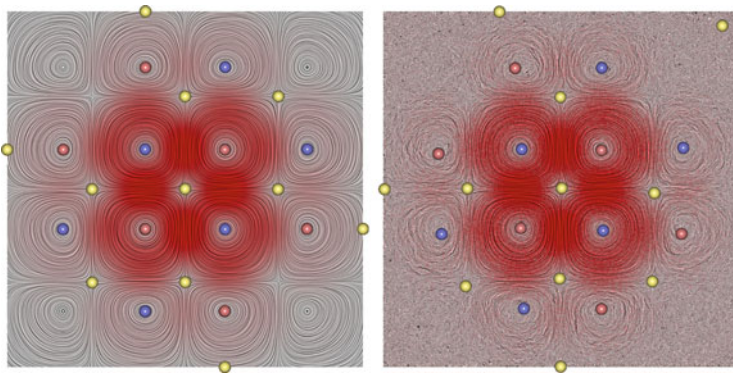
Fig. 3 A synthetic divergence-free vector field is depicted using a LIC image colored by magnitude. The critical points of $V^{\phi}_{k_n - 11}$ are shown. The saddles, CW-centers, and CCW-centers are depicted as *yellow*, *blue*, and *red* spheres. *Left*: the original smooth vector field. *Right*: a noisy measurement of the field depicted on the left
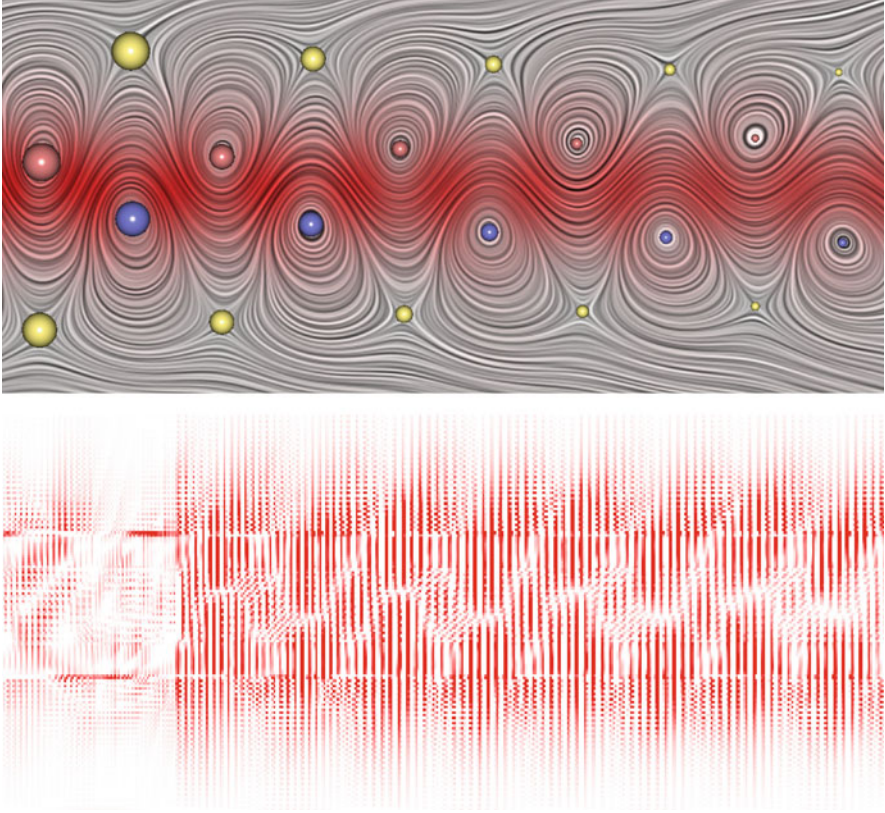
Fig. 4 *Top*: A quasi-divergence-free vector field of the flow behind a circular cylinder is depicted using a LIC image colored by magnitude. The saddles, CW-centers, and CCW-centers are depicted as *yellow*, *blue*, and *red* spheres and are scaled by our importance measure. *Bottom*: the divergence of the data set is shown using a colormap (*white*: zero divergence, *red*: high divergence)

The data set exhibits the well-known Kármán vortex street of alternating clockwise and counter-clockwise rotating vortices. This structure is extracted well by our algorithm. The strength of the vortices decreases the further they are from the cylinder on the left. This physical property is reflected well by our importance measure for critical points in divergence-free vector fields.

6 Conclusion

We presented an algorithm for the extraction of critical points in 2D divergence-free vector fields. In contrast to previous work this algorithm is provably consistent in the sense of Morse theory for divergence-free vector fields as presented in Sect. 3.

It also allows for a consistent simplification of the set of critical points which enables the analysis of noisy data as illustrated in Fig. 3. The computed importance measure has a physical relevance as shown in Fig. 4, and allows to discriminate between dominant and spurious critical points in a data set. By combinatorially enforcing the gradient vector field property we are able to directly deal with data sets with only near zero divergence (see Fig. 4, bottom).

The only step of our algorithm that is not combinatorial is the Helmholtz-Hodge decomposition which is necessary for surfaces of higher genus to get the harmonic-free part of the vector field. It would therefore be interesting to investigate the possibility of a purely combinatorial Helmholtz-Hodge decomposition. Alternatively, one could try to develop a computational discrete Morse theory for divergence-free vector fields containing a harmonic part.

References

1. Bauer, U., Lange, C., Wardetzky, M.: Optimal topological simplification of discrete functions on surfaces. arXiv:1001.1269v2 (2010)
2. Chari, M.K.: On discrete Morse functions and combinatorial decompositions. Discrete Math. **217**(1-3), 101–113 (2000)
3. Chen, G., Mischaikow, K., Laramee, R., Pilarczyk, P., Zhang, E.: Vector field editing and periodic orbit extraction using Morse decomposition. IEEE Trans. Visual. Comput. Graph. **13**, 769–785 (2007)
4. Chen, G., Mischaikow, K., Laramee, R.S., Zhang, E.: Efficient morse decomposition of vector fields. IEEE Trans. Visual. Comput. Graph. **14**(4), 848–862 (2008)
5. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. Discrete Comput. Geom. **28**, 511–533 (2002)
6. Forman, R.: Combinatorial vector fields and dynamical systems. Mathematische Zeitschrift **228**(4), 629–681 (1998)
7. Forman, R.: Morse theory for cell complexes. Adv. Math. **134**, 90–145 (1998)
8. Gyulassy, A.: Combinatorial Construction of Morse-Smale Complexes for Data Analysis and Visualization. PhD thesis, University of California, Davis (2008)
9. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge, U.K. (2002)
10. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. IEEE Comput. **22**(8), 27–36 (1989)
11. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. Computer **22**(8), 27–36 (1989)
12. Helmholtz, H.: Über integrale der hydrodynamischen gleichungen, welche den wirbelbewegungen entsprechen. J. Reine Angew. Math. **55**, 25–55 (1858)
13. King, H., Knudson, K., Mramor, N.: Generating discrete Morse functions from point data. Exp. Math. **14**(4), 435–444 (2005)
14. Klein, T., Ertl, T.: Scale-space tracking of critical points in 3d vector fields. In: Helwig Hauser, H.H., Theisel, H. (eds.) Topology-based Methods in Visualization, Mathematics and Visualization, pp. 35–49. Springer, Berlin (2007)

15. Laramee, R.S., Hauser, H., Zhao, L., Post, F.H.: Topology-based flow visualization, the state of the art. In: Helwig Hauser, H.H., Theisel, H. (eds.) Topology-based Methods in Visualization, Mathematics and Visualization, pp. 1–19. Springer, Berlin (2007)
16. Lewiner, T.: Geometric discrete Morse complexes. PhD thesis, Department of Mathematics, PUC-Rio, 2005. Advised by Hélio Lopes and Geovan Tavares.
17. Milnor, J.: Morse Theory. Princeton University Press, Princeton (1963)
18. Noack, B.R., Schlegel, M., Ahlborn, B., Mutschke, G., Morzyński, M., Comte, P., Tadmor, G.: A finite-time thermodynamics of unsteady fluid flows. J. Non-Equilibr. Thermodyn. **33**(2), 103–148 (2008)
19. Panton, R.: Incompressible Flow. Wiley, New York (1984)
20. Petronetto, F., Paiva, A., Lage, M., Tavares, G., Lopes, H., Lewiner, T.: Meshless Helmholtz-Hodge decomposition. Trans. Visual. Comput. Graph. **16**(2), 338–342 (2010)
21. Polthier, K., Preuß, E.: Identifying vector field singularities using a discrete Hodge decomposition. pp. 112–134. Springer, Berlin (2002)
22. Reininghaus, J., Günther, D., Hotz, I., Prohaska, S., Hege, H.-C.: TADD: A computational framework for data analysis using discrete Morse theory. In: Fukuda, K., van der Hoeven, J., Joswig, M., Takayama, N. (eds.) Mathematical Software – ICMS 2010, volume 6327 of Lecture Notes in Computer Science, pp. 198–208. Springer, Berlin (2010)
23. Reininghaus, J., Hotz, I.: Combinatorial 2d vector field topology extraction and simplification. In: Pascucci, V., Tricoche, X., Hagen, H., Tierny, J. (eds.) Topological Methods in Data Analysis and Visualization, Mathematics and Visualization, pp. 103–114. Springer, Berlin (2011)
24. Reininghaus, J., Löwen, C., Hotz, I.: Fast combinatorial vector field topology. IEEE Trans. Visual. Comput. Graph. **17** 1433–1443 (2011)
25. Robins, V., Wood, P.J., Sheppard, A.P.: Theory and algorithms for constructing discrete morse complexes from grayscale digital images. IEEE Trans. Pattern Anal. Mach. Learn. **33**(8), 1646–1658 (2011)
26. Shonkwiler, C.: Poincaré duality angles for Riemannian manifolds with boundary. Technical Report arXiv:0909.1967, Sep 2009. Comments: 51 pages, 6 figures.
27. Tong, Y., Lombeyda, S., Hirani, A.N., Desbrun, M.: Discrete multiscale vector field decomposition. In: ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH, volume 22 (2003)
28. Tricoche, X., Scheuermann, G., Hagen, H.: Continuous topology simplification of planar vector fields. In: VIS '01: Proceedings of the conference on Visualization '01, pp. 159–166. IEEE Computer Society, Washington, DC, USA (2001)
29. Tricoche, X., Scheuermann, G., Hagen, H., Clauss, S.: Vector and tensor field topology simplification on irregular grids. In: Ebert, D., Favre, J.M., Peikert, R. (eds.) VisSym '01: Proceedings of the symposium on Data Visualization 2001, pp. 107–116. Springer, Wien, Austria, May 28–30 (2001)
30. Weinkauf, T.: Extraction of Topological Structures in 2D and 3D Vector Fields. PhD thesis, University Magdeburg (2008)
31. Weinkauf, T., Theisel, H., Shi, K., Hege, H.-C., Seidel, H.-P.: Extracting higher order critical points and topological simplification of 3D vector fields. In: Proceedings IEEE Visualization 2005, pp. 559–566. Minneapolis, U.S.A., October 2005

Efficient Computation of a Hierarchy of Discrete 3D Gradient Vector Fields

David Günther, Jan Reininghaus, Steffen Prohaska, Tino Weinkauf, and Hans-Christian Hege

1 Introduction

The analysis of three dimensional scalar data has become an important tool in scientific research. In many applications, the analysis of topological structures – the critical points, separation lines and surfaces – are of great interest and may help to get a deeper understanding of the underlying problem. Since these structures have an extremal characteristic, we call them *extremal structures* in the following.

The extremal structures have a long history [2, 14]. Typically, the critical points are computed by finding all zeros of the gradient, and can be classified into minima, saddles, and maxima by the eigenvalues of their Hessian. The respective eigenvectors can be used to compute the separation lines and surfaces as solutions of autonomous ODEs. For the numerical treatment of these problems we refer to Weinkauf [22].

One of the problems that such numerical algorithms face is the discrete nature of the extremal structures. For example, the type of a critical point depends on the signs of the eigenvalues. If the eigenvalues are close to zero, the determination of the type is ill-posed and numerically challenging. Depending on the input data, the resulting extremal structure may therefore strongly depend on the algorithmic parameters and numerical procedures. From a topological point of view, this can be quite problematic. Morse theory relates the extremal structure of a generic function to the topology of the manifold, e.g., by the Poincaré-Hopf Theorem or by the

D. Günther (✉), J. Reininghaus, S. Prohaska, H.-C. Hege
Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
e-mail: david.guenther@zib.de; reininghaus@zib.de; prohaska@zib.de; hege@zib.de

T. Weinkauf
Courant Institute of Mathematical Sciences, New York University, 715 Broadway,
New York, NY 10003, U.S.A
e-mail: weinkauf@courant.nyu.edu

strong Morse inequalities [15]. The topology of the manifold restricts the set of the admissible extremal structures.

Another problem is the presence of noise, for example due to the imaging process, or sampling artifacts. Both can create fluctuations in the scalar values that may create additional extremal structures, which are very complex and hard to analyze, in general. A distinction between important and spurious elements is thereby crucial.

To address these problems, one may use the framework of discrete Morse theory introduced by Forman who translated concepts from continuous Morse theory into a discrete setting for cell complexes [5]. A gradient field is encoded in the combinatorial structure of the cell complex, and its extremal structures are defined in a combinatorial fashion. A finite cell complex can therefore carry only a finite number of combinatorial gradient vector fields, and their respective extremal structures are always consistent with the topology of the manifold.

The first computational realization of Forman's theory was presented by Lewiner et al. [12, 13] to compute the homology groups of 2D and 3D manifolds. In this framework, a sequence of consistent combinatorial gradient fields can be computed such that the underlying extremal structures become less complex with respect to some importance measure. The combinatorial fields are represented by hypergraphs and hyperforests, which allow for a very compact and memory efficient representation of the extremal structure. However, the framework is only applicable to relatively small three dimensional data sets since the construction of the sequence requires several graph traversals. This results in a non-feasible running time for large data sets. Recently, several alternatives for the computation of a discrete Morse function were proposed, for example by Robins et al. [18] and King et al. [11].

An alternative approach to extract the essential critical points and separation lines was proposed by Gyulassy [7]. His main idea is to construct a single initial field and extract its complex extremal structures by a field traversal. To separate spurious elements from important ones, the extremal structures are then directly simplified. One advantage of this approach is a very low running time. One drawback is that certain pairs of critical points, i.e., the saddle points, may be connected among each other arbitrarily often by saddle connectors [21]. This can result in a large memory overhead [8] since the connectors as well as their geometric embedding need to be stored separately. Note that the reconstruction of a combinatorial gradient vector field based only on a set of critical points and their separation lines is challenging.

In this work, we construct a nested sequence of combinatorial gradient fields. The extremal structures are therein implicitly defined, which enables a memory-efficient treatment of these structure. Additionally, the complete combinatorial flow is preserved at different levels of detail, which allows not only the extraction of separation surfaces, but may also be useful for the analysis of 3D time-dependent data as illustrated by Reininghaus et al. [17] for 2D.

The computation of our sequence is based on the ideas of Reininghaus et al. [16]. A combinatorial gradient field is represented by a Morse matching in a derived cell graph. In this paper, we focus on scalar data given on a 3D structured grid.

Although the computation of a sequence of Morse matchings is a global problem, an initial Morse matching can be computed locally and in parallel. We use an OpenMP implementation of the *ProcessLowerStar*-algorithm proposed by Robins et al. [18] to compute this initial matching. The critical points in this matching correspond one-to-one to the changes of the topology of the lower level cuts of the input data.

As mentioned earlier, the presence of noise may lead to a very complex initial extremal structure. The objective of this paper is to efficiently construct a nested sequence of Morse matchings such that every element of this sequence is topologically consistent, and the underlying extremal structures become less complex in terms of number of critical points. The ordering of the sequence is based on an importance measure that is closely related to the persistence measure [4, 24], and is already successfully used by Lewiner [12] and Gyulassy [7]. This measure enables the selection of a Morse matching with a prescribed complexity of the extremal structure in a very fast, almost interactive post-processing step. The critical points and the separation lines and surfaces are then easily extracted by collecting all unmatched nodes in the graph and a constrained depth-first search starting at these nodes.

The rest of the paper is organized as follows: in Sect. 2, we formulate elements of discrete More theory in graph theoretical terms. In Sect. 3, we present our new algorithm for constructing a hierarchy of combinatorial gradient vector fields. In Sect. 4, we present some examples to illustrate the result of our algorithm and its running time.

2   Computational Discrete Morse Theory

This section begins with a short introduction to discrete Morse theory in a graph theoretical formulation. We then recapitulate the optimization problem that results in a hierarchy of combinatorial gradient vector fields representing a 3D image data set. For simplicity, we restrict ourselves to three dimensional scalar data given on the vertices of a uniform regular grid. The mathematical theory for combinatorial gradient vector fields, however, is defined in a far more general setting [5].

2.1   *Cell Graph*

Let C denote a finite regular cell complex [9] defined by a 3D grid. In this paper, we call a cell complex *regular* if the boundary of each d-cell is contained in a union of $(d-1)$-cells. The cell graph $G = (N, E)$ encodes the combinatorial information
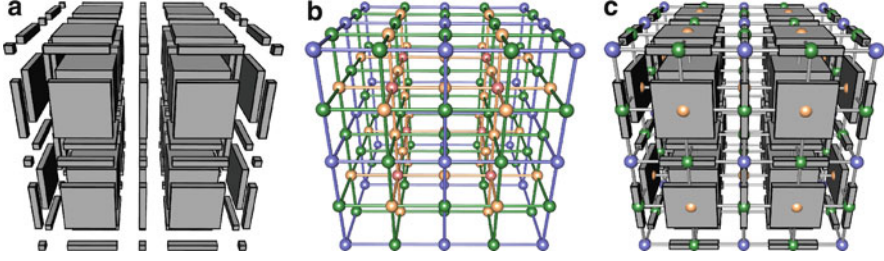
Fig. 1 Illustration of a cell complex and its derived cell graph. (**a**) shows the cells of a $2 \times 2 \times 2$ uniform grid in an exploded view. A single voxel is represented by eight 0-cells, twelve 1-cells, six 2-cells, and one 3-dimensional cell. These cells and their boundary relation define the cell complex. (**b**) shows the derived cell graph. The nodes representing the 0-, 1-, 2-, and 3-cells are shown as *blue*, *green*, *yellow* and *red spheres* respectively. The adjacency of the nodes is given by the boundary relation of the cells. The edges are colored by the lower dimensional incident node. (**c**) shows the cell complex and the cell graph to illustrate the neighborhood relation of the cells

contained in C. The nodes N of the graph consist of the cells of the complex C and each node u^p is labeled with the dimension p of the cell it represents. The scalar value of each node is also stored. Higher dimensional nodes are assigned the maximal scalar value of the incident lower dimensional nodes. The edges E of the graph encode the neighborhood relation of the cells in C. If the cell u^p is in the boundary of the cell w^{p+1}, then $e^p = \{u^p, w^{p+1}\} \in E$. We label each edge with the dimension of its lower dimensional node. An illustration of a cell complex and its graph is shown in Fig. 1. Note that the node indices, their adjacence and their geometric embedding in $\mathbb{R}^3$ are given implicitly by the grid structure.

2.2 Morse Matchings

A subset of pairwise non-adjacent edges is called a *matching $M \subset E$*. Using these definitions, a *combinatorial gradient vector field V* on a regular cell complex C can be defined as a certain acyclic matching of the cell graph G [3]. The set of combinatorial gradient vector fields on C is given by the set of these matchings, i.e., the set of *Morse matchings $\mathcal{M}^\phi$* of the cell graph G. An illustration of a 2D Morse matching is shown in Fig. 2a.

2.3 Extremal Structures

We now define the extremal structures of a combinatorial gradient vector field V in G. The unmatched nodes are called *critical nodes*. If u^p is a critical node, we say that the critical node has index p. A *critical node* of index p is called
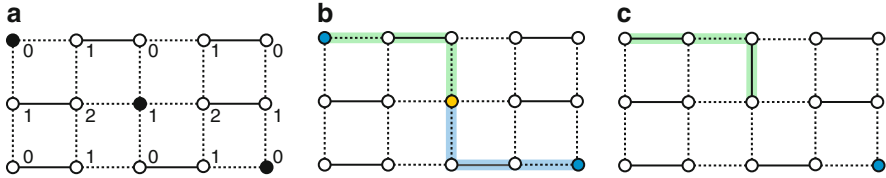
Fig. 2 Depiction of algorithm *constructHierarchy*. Image (**a**) shows a $2D$ Morse matching M. The matched and unmatched edges of the cell graph G are depicted as *solid* and *dashed lines* respectively. The unmatched nodes of G are shown as *black dots*. Each node of G is labeled by its dimension. Image (**b**) shows the two minima (*blue dots*) and the saddle (*yellow dot*) as well as the only two possible augmenting paths (*blue and green stripes*) in M. Image (**c**) shows the augmentation of M along the left (*green*) path. The start- and endnode of this path are now matched and not critical anymore. A single minimum (*blue dot*) remains in M

minimum ($p = 0$), 1-saddle ($p = 1$), 2-saddle ($p = 2$), or maximum ($p = 3$). A combinatorial *p-streamline* is a path in the graph whose edges are of dimension p and alternate between $V \subset E$ and its complement $E \setminus V$. In a Morse matching, there are no closed p-streamlines. This defines the acyclic constraint for Morse matchings. A p-streamline connecting two critical nodes is called a *p-separation line*. A *p-separation surface* is given by all combinatorial 1-streamlines that emanate from a critical point of index p. The extremal structures give rise to a Morse-Smale complex that represents the topological changes in the level sets of the input data. Since we have assigned the maximal value to higher dimensional cells, there are no saddles with a scalar value smaller or greater than their connected minima or maxima respectively.

2.4 Optimization Problem

The construction of a hierarchy can be formulated as an optimization problem [16]. Given edge weights $\omega : E \to \mathbb{R}$, the objective is to find an acyclic matching $V_k \in \mathscr{M}^\phi$ such that

$$V_k = \underset{M \in \mathscr{M}^\phi, |M|=k}{\arg\max} \omega(M). \tag{1}$$

However, (1) becomes an NP-hard problem in the case of 3D manifolds [10]. We therefore only use (1) to guide our algorithmic design to construct a nested sequence of combinatorial gradient vector fields $\mathscr{V} = (V_k)_{k=k_0,\ldots,k_n}$. For each k, we are looking for the smallest fluctuation to get a representation of our input data at different levels of detail. Note that this proceeding differs from the homological persistence approach introduced by Edelsbrunner et al. [4]. There are persistence pairs in $3D$ that cannot be described by a sequence $\mathscr{V}$ as shown in a counterexample by Bauer et al. [1].

3 Algorithm

In this section, we describe the construction of a sequence of combinatorial gradient vector fields. The construction consists of two steps. In the first step, an initial Morse matching is computed. The matching represents the fine-grained flow of the input data. In the second step, the initial matching is iteratively simplified by removing the smallest fluctuation in every iteration. The simplification is done by computing the p-separation line S representing this fluctuation in a given matching V_ℓ. A p-separation line, which is connecting two critical points, is an augmenting path since it is alternating and its start- and endnode are not matched. We can then produce a larger matching $V_{\ell+1}$ by taking the symmetric difference

$$V_{\ell+1} = V_\ell \bigtriangleup S. \tag{2}$$

Equation (2) is called *augmenting* the matching. The simplification stops if the matching can not be augmented anymore. This final result represents the gradient field with the coarsest level of detail.

3.1 Initial Matching

To compute the initial matching V_{k_0}, we use the algorithm *ProcessLowerStar* [18]. *ProcessLowerStar* computes a valid Morse matching by finding pairs in the lower star of each 0-node in lexicographic descending order. Since the decomposition of a cell graph in its lower stars is a disjoint decomposition, each lower star can be processed in parallel. The assumption in *ProcessLowerStar* is that the scalar values are distinct. To fulfill this requirement, we use the same idea as Robins et al. [18]. Two 0-nodes in a lower star with the same scalar values are differentiated by their index. If the enumeration of the 0-nodes in G is linear, this correlates to a linear ramp with an infinitesimal small η.

3.2 Computing the Hierarchy

In the following we describe the construction of a sequence of Morse matchings $\mathcal{V}$. See Algorithm 1 and Fig. 2 for a depiction of it. The main idea is to compute the p-separation line with the smallest weight that emanates from a saddle and allows for an augmentation of the Morse matching. While the computation of the 0- and 2-separation lines is straight forward, special attention needs to be taken for the computation of 1-separation lines since they can merge and split in the combinatorial setting.

Algorithm 1: constructHierarchy

Input: *initial matching* V_{k_0}, *cell graph G*
Output: *hierarchy*
1: *hierarchy* ← *nil*
2: *saddleQueue* ← *initQueue*(*G*)
3: **while** *saddleQueue* ≠ ∅ **do**
4:     *s* ← *saddleQueue.pop*()
5:     **if** *isCritical*(*s*) **then**
6:         [*cancelPartner*, *augPath*] ← *getUniquePairing*(*s.idx*)
7:         **if** *cancelPartner* **then**
8:             *weight* ← *getWeight*(*s.idx*, *cancelPartner*)
9:             **if** *weight* < *saddleQueue.top*().*weight* **then**
10:                 *updateMatching*(*augPath*)
11:                 *hierarchy.append*(*augPath*)
12:             **else**
13:                 *saddleQueue.push*(*s.idx*, *weight*)

We start with the initial matching V_{k_0} as described above. In the first step, the priority queue is initialized by the function *initQueue* (line 2). This function collects all unmatched 1- and 2-nodes and computes the weight of these nodes. The weight is given by the smallest difference in the scalar values of a saddle and its neighbors [16]. *initQueue* uses basically the same functionality as the function *getUniquePairing*, which is described subsequently. After the queue is initialized, the first saddle *s* of the queue, i.e., the element with the smallest weight, is taken (line 4) and checked whether it is still critical (line 5). This is necessary since previous simplification steps may have affected *s*. Then, the function *getUniquePairing* computes the cancel partner as well as the augmenting path that connects this node with *s* (line 6). If the saddle *s* is connected to every neighbor by multiple paths, then we can not cancel this saddle since a closed combinatorial streamlines would be created (line 7). Otherwise, we compute the weight of *s* and its cancel partner and test whether it is smaller than the weight of the next element in the queue (line 8 and 9). This is necessary since previous simplification steps may have affected the connectivity of *s*. If the weight is smaller, it represents the smallest fluctuation at this time, and we can augment the matching along the path (line 10). This results in a simplified combinatorial gradient field where the saddle node *s* and its cancel partner are no longer critical. Since the augmentation of a matching along an augmenting path never creates new critical nodes, the complexity of the underlying extremal structure is reduced. The path is finally stored to be able to restore this specific level of detail (line 11). We reinsert the saddle *s* with the new weight (line 13) if the weight is greater.

The main computational effort lies in the computation of the best pairing that contains a uniquely defined connection. Algorithm 2 and Fig. 3 show how this can be achieved efficiently. Let *s* be an unmatched 1- or 2-node. In the first step, the two 0- or 2-separation lines – the paths that connect a saddle node with at most two 0-nodes or 3-nodes – are computed. We take the two 0- or 2-edges incident to *s*

Algorithm 2: getUniquePairing

  Input: *saddle s*
  Output: *cancelPartner*, *augmentingPath*
  1: *cancelPartner ← nil, augmentingPath ← nil, isCircle ← false, weight ← ∞*
  2: [*firstLink*, *secLink*] ← *getLinkToExtrema*(*s*)
  3: [*firstPath*, *secPath*] ← *integrateSeparationLine*(*s*, [*firstLink*, *secLink*])
  4: **if** *getEndNode*(*firstPath*) ≠ *getEndNode*(*secPath*) **then**
  5:     [*cancelPartner*, *augmentingPath*] ← *getBestWeight*(*firstPath*, *secPath*)
  6:     *weight ← getWeight*(*s*, *cancelPartner*)
  7: [*surface*, *saddles*] ← *integrateSeparationSurface*(*s*)
  8: *sort*(*saddles*)
  9: **for all** *n* ∈ *saddles* **do**
 10:     **if** *n.weight < weight* **then**
 11:       [*isCircle*, *line*] ← *checkMultiplePairing*(*surface*, *n.idx*)
 12:       **if** *isCircle = false* **then**
 13:         *weight ← n.weight, cancelPartner ← n.idx*
 14:         *augmentingPath ← line*
 15:         **return**
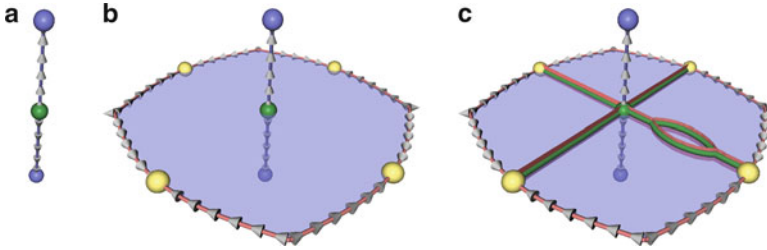
Fig. 3 Illustration of algorithm *getUniquePairing*. In the first step (**a**), the two 1-separation lines (*blue lines*) starting from a 1-saddle (*green sphere*) are integrated. Both end in distinct minima (*blue spheres*), which would allow for an augmentation along one of these lines. The combinatorial flow restricted to the separation lines is indicated by *arrows*. In the second step (**b**), the separation surface (*blue surface*) is integrated using a depth first search. The surface ends in 2-separation lines (*red lines*) that emanate from 2-saddles (*yellow spheres*). For each of these 2-saddles the intersection of their separation surface and the surface emanating from the 1-saddle is computed in the third step (**c**). The intersection is depicted by *red stripes*. The resulting saddle connectors, i.e., the 1-separation lines, are shown as *green lines*. The right 2-saddle is connected twice with the 1-saddle. An augmentation of the matching along one of these lines would result in a closed 1-streamline. This saddle is therefore not a valid candidate for a cancellation. From the remaining 2-saddles and the two minima the critical node is chosen that has the smallest weight with respect to the 1-saddle

(line 2) and follow the combinatorial gradient field until an unmatched node is found. This is done by the function *integrateSeparationLine* (line 3). Note that these separation lines are uniquely defined if we start at a saddle. Multiple lines can merge but they can not split. We need to check whether these two paths end in the same minimum or maximum (line 4). If they do, an augmentation along one of these paths would create a closed streamline, which are not allowed in combinatorial gradient

---

Algorithm 3: getUniqueSaddleConnector

Input: *separation surface sepSurf*, *saddle s*
Output: *sepLine*, *isCircle*
1: *sepLine* ← *nil*,
2: *queue* ← *nil*, *queue.push(s)*
3: **while** *queue* ≠ ∅ **do**
4:     *curNode* ← *queue.pop()*, *numNeighbors* ← 0
5:     *nodeList* ← *getAllNeighborsInSurface(curNode, sepSurf)*
6:     **for all** *m* ∈ *nodeList* **do**
7:         **if** *isVisted[m]* = *false* **then**
8:             **if** *numNeighbors* > 1 **then**
9:                 *isCircle* ← *true*
10:                **return**
11:            **else**
12:                *queue.push(m)*, *sepLine.push(getLink(m, curNode))*
13:                *isVisted[m]* ← *true*
14:                *numNeighbors* ← *numNeighbors* + 1

---

fields. If the two endnodes are distinct, we choose the one with the smallest weight and take the corresponding path as augmenting path (line 5 and 6).

In the second step, we investigate the connectivity of *s* with complementary saddle nodes. The 1-separation lines that connect these saddles are also called saddle connectors [21], and are defined by the intersection of the complementary separation surfaces. In contrast to 0- and 2-separation lines, these lines can split and merge. In previous work of Lewiner [12], this property results in a non-feasible running time, and in the work of Gyualssy [7], it induces a large memory consumption. The second part of Algorithm 2 and Algorithm 3 show a memory and running time efficient alternative.

Given the saddle *s*, we integrate the separation surface using a depth-first search (line 7). This is done by *integrateSeparationSurface*. Note that the integration only follows the 1-streamlines, i.e., the 1-paths that alternate between the current matching and its complement. Since the boundary of a separation surface consists of separation lines, the integration will terminate at these lines. The 1- and 2- nodes describing these lines are already matched and hinder a further flooding. The result of *integrateSeparationSurface* is a list of 1- and 2-nodes representing the separation surface. Additionally, a list of the complementary saddles is returned. We sort these saddles by their weight to *s* (line 8) and test them in ascending order (line 9). Since the objective is to remove the smallest fluctuation, we are looking for a saddle partner with a smaller weight than *s* has with its uniquely connected minima or maxima (line 10). If there is such a partner, we check whether there are multiple connections between these two saddles by calling *getUniqueSaddleConnector* (line 11). If the connection is unique we use it as an augmenting path and return (line 13, 14 and 15).

In the discrete Morse setting of Forman's theory, saddle connectors can merge and split. This property prohibits a direct walk starting at a saddle as we have done

for the 0- and 2-separation lines. Saddle connectors could be computed by definition as the intersection of the two corresponding separation surfaces [21], but this would result in a infeasible running time. Instead, we compute the intersection directly using the function *getUniqueSaddleConnector*, shown in Algorithm 3.

Consider a set of 1- and 2-nodes representing a separation surface, and a saddle s in the boundary of this surface. We first push s in a queue (line 2). This queue will allow the traversal of the saddle connector. For the first element of the queue, we collect all neighboring 1- and 2-nodes in the node list given by the separation surface (line 4 and 5). Note that the saddle connector is a 1-streamline and its edges must alternate between the matching and its complement. This is achieved by the function *getAllNeighborsInSurface*. The main idea is now to check for split events in the intersection. If there is such an event, we know that there are multiple connections between the two saddles since by definition the intersection always ends in the complementary saddle. We test each of these nodes if they were already visited (line 6 and 7). In order to check for split events, we need to count the number of possible extensions of the saddle connector. If there are more than one, the algorithm returns with a boolean indicating multiple connections (line 8, 9 and 10). If this is not the case, the current node is an extension of the saddle connector. The node is added to the queue and the corresponding link to the saddle connector. The number of possible extensions is increased by one (line 12, 13 and 14). The loop ends in the other saddle, and the links describing the saddle connector are in sorted order.

3.3 Extraction of Extremal Structures

Given a nested sequence of combinatorial gradient vector fields $\mathscr{V} = (V_k)_{k=k_0,\ldots,k_n}$, an arbitrary element of the sequence can be restored as follows: first we take the coarsest possible field V_{k_n}. This is the final result of Algorithm 1. Note that this field can be efficiently represented by a boolean vector whose size is given by the number of edges in G. Then, this field is iteratively augmented along the augmenting paths computed in Algorithm 1 in reverse order $(V_{k_{n-1}},\ldots,V_{k_1})$. The augmentation of a field V_ℓ along an alternating path p_ℓ is given by the symmetric difference $V_{\ell-1} = V_\ell \triangle p_\ell$. In contrast to (2) this augmentation increases the number of critical nodes by two. The augmentation stops if the desired number of critical nodes is achieved or the weight of the last augmenting path corresponds to a prescribed threshold.

For a certain level in the hierarchy $\mathscr{V}$, the critical nodes are computed by collecting all unmatched nodes. From each of the collected 1- and 2-nodes, the 0- and 2-separation lines are computed by following the combinatorial flow. The separation surfaces are restored by a depth-first search similar as is used in Algorithm 2. For the computation of the saddle connectors, we can use Algorithm 3 in a slightly modified version. Instead of returning when a split event was found, a new line is started. The geometric embedding is given by the grid structure

of the input data. Note that the extremal structures can not be easily updated incrementally.

3.4 Memory Consumption

For the construction of the hierarchy, a boolean vector, whose size is given by the number of edges in G, is needed to represent the current matching. The number of nodes in the cell graph is eight times the number of vertices in the input grid. The number of edges in G is therefore bounded by 24 times the number of vertices. Since the size of a boolean is 1/32th of a single precision number we need a factor of 0.75 of the input data to represent the matching. Three additional boolean vectors of size of number of nodes are necessary for the surface integration, its intersection, and the node matching. The total factor is therefore 1.5 of the input data. Robins proved that the critical points are in a one-to-one correspondence to the topological changes in the lower level sets [18]. Since (2) only decreases the number of critical nodes, the size of the priority queue is given by the number of critical points in the input field. The theoretical maximal memory consumption for separation surfaces is bounded by the number of 1- and 2-nodes in G.

Table 1 Running times and memory consumption for six data sets of varying dimensions. The computation of the initial matching with 12 cores and, as reference, for 1 core is shown in the second column. The resulting speed up factor is shown in the third column. The running time for a 5% and a complete simplification, and the number of levels in the hierarchies are shown in the fourth and fifth column. The peak memory consumption and the memory factor for a full simplification including the augmenting paths are shown in the sixth and seventh column

Data set (Size)	Initial matching 12 cores (1 core)	Speed up	5% Simplification Complete hierarchy	$\mathcal{V}_{5\%}$ $\mathcal{V}$	Peak memory	Memory factor
Neghip	6 s	10.3	11 s	4,974	1 MB	1
64^3	(1 min 2 s)		11 s	5,023		
Hydrogen	51 s	11.6	2 min 13 s	87,821	8 MB	1
128^3	(9 min 53 s)		2 min 13 s	87,825		
Aneurism	7 min 02 s	11.54	15 min 45 s	38,542	107 MB	1.59
256^3	(81 min 12 s)		21 min 39 s	48,561		
Beetle	18 min 43 s	11.46	34 min 26 s	309,290	260 MB	1.52
$416^2 \times 247$	(214 min 26 s)		41 min 19 s	321,222		
Benzene	30 min 46 s	–	33 min 1 s	92	392 MB	1.51
401^3			33 min 7 s	123		
Synthetic	8 h 26 min 19 s	–	8 h 45 min 22 s	203	6.3 GB	1.51
$1,024^3$			8 h 49 min 16 s	243		
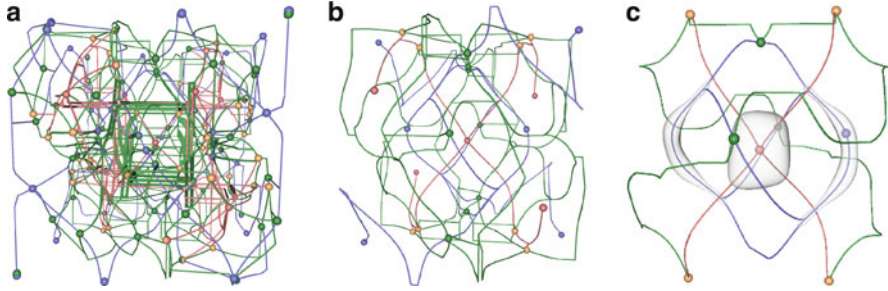
Fig. 4 This image shows the critical points and the *p*-separation lines of a synthetic example for different levels of detail. Minima, 1-saddles, 2-saddles and maxima are depicted as *blue*, *green*, *yellow* and *red spheres* respectively. The *p*-separation lines are shown as *blue* ($p = 0$), *green* ($p = 1$) and *red* ($p = 2$) *lines*. Image (**a**) shows the initial Morse matching V_{k_0} whereas (**b**) and (**c**) show the level V_{k_n-13} and V_{k_n-4}. The isosurface (*grey*) in c) illustrates the most dominant minima and maxima regions. The hierarchy consists of 243 levels

4 Examples

In the following, we present some examples to illustrate our method. All experiments were done on an AMD Opteron 6174 CPU. To compute the initial matching, we implemented an OpenMP version of *ProcessLowerStar*. Table 1 shows the running time and memory consumption for different 3D data sets. The neghip, hydrogen and aneurism are provided by *The Volume Library* [19] while the beetle data set is provided by Gröller et al. [6]. We give the running time for the computation of the initial matching with 12 cores and 1 core, respectively, and the corresponding speed up factor. Besides computing the complete hierarchization, it is in some cases sufficient to compute only a subsequence of V in order to remove only the most spurious/noisy extremal structures. Therefore, we also give the computation time of the algorithm for a 5% simplification, i.e., until the weight of the last augmenting path corresponds to 5% of the data range. The corresponding number of hierarchy levels is given as well. The memory consumption is measured by observing the peak memory usage during computation. This includes also the augmenting paths. The memory factor relates the consumption to the file size (single point precision). Figure 4 shows the extremal structures for different levels of detail of a synthetic example. The running time and memory consumption is also given in Table 1.

The speed up factor is nearly optimal and scales with the dimensions of the data set. The construction time of $\mathscr{V}$ for the complex aneurism data set was approximately 21 min, which correlates to the work of Gyulassy et al. [8] with a reasonable valence parameter. This example shows also that the topological complexity of the initial field influences the running time. For simple data sets as the neghip or hydrogen there is nearly no difference in running time between a 5% and a full simplification. The overall running time and the practical memory consumption, which is less than a factor of two of the input data, allows for the analysis of large data with an appropriate topological complexity.
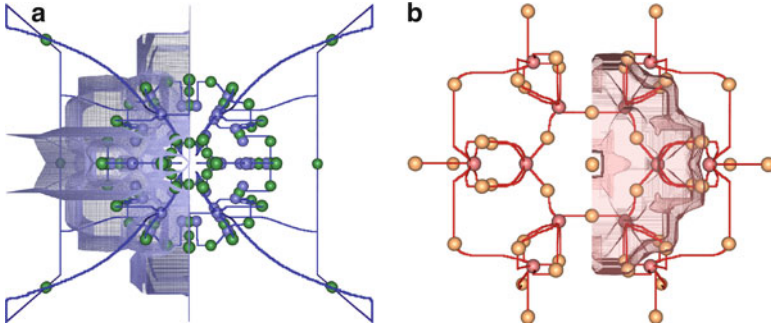
Fig. 5 Extremal structures of the electrostatic field of a benzene molecule for V_{k_n-90} with 181 critical points. (**a**) shows the minimal structures: the 48 minima (*blue spheres*), 78 1-saddles (*green spheres*), 0-separation lines (*blue lines*) and the 1-separation surface (*blue surface*). (**b**) shows the maximal structures: the 12 maxima (*red spheres*), 43 2-saddles (*yellow spheres*), 2-separation lines (*red lines*) and the 2-separation surface (*red surface*). Ude to symmetry, only one half of the separation surfaces is shown. Note that 1-separation surfaces separate the flow given by the 0-streamlines, while 2-separation surfaces separate 2-streamlines. Triangulating the 2-nodes of 1-separation surfaces therefore does not necessarily lead to closed surfaces in contrast to 2-separation surfaces
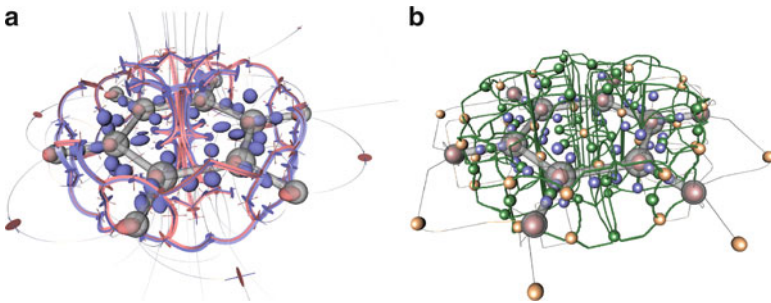
Fig. 6 Comparison of combinatorial and continuous extremal structures for the electrostatic field around a benzene molecule. Image (**a**) shows smooth extremal structures extracted as in [21]. The minima and the maxima are depicted as *blue* and *red spheres* while the 1- and 2-saddles are shown as *blue* and *red disks* respectively. The saddle connectors are shown as *blue-red stripes*. *Gray illuminated lines* represent streamlines emanating from the saddles. Image (**b**) shows combinatorial extremal structures. The minima, 1- and 2-saddles, and the maxima are represented by *blue*, *green*, *yellow* and *red spheres* respectively. The saddle connectors are shown as *green lines*. *Gray illuminated lines* depict the 2-separation lines emanating from the 2-saddles. *Gray surfaces* depict the carbon and the hydrogen atoms and their bonds

4.1 Comparison of Continuous and Combinatorial Extremal Structures

Figures 5 and 6 visualize the extremal structures of the electrostatic potential around the benzene molecule. This data set has been analyzed by Theisel et al. [21] using

numerical methods, and we use their results for a side-by-side comparison of continuous and combinatorial structures. To achieve comparable results, we chose the hierarchy level V_{k_n-90} where we have the same number of 1- and 2-saddles as in the continuous case. The data set is sampled on a 401^3 regular grid using the fractional charges method [20]. The running time is shown in Table 1; the extraction of the critical points and the p-separation lines for an arbitrary element of $\mathscr{V}$ took at most 20 s whereas the separation surfaces took at most 60 s. Figure 5 shows our combinatorial result from a top view. Note how the regularity of the underlying data set has been perfectly captured. This poses a challenge for numerical algorithms, since guarantees about finding all critical points can usually not be given. The side-by-side comparison of the continuous and the combinatorial extraction results is shown in Fig. 6.

We make the following observations: First, the continuous version is not only visually more pleasing, but it better communicates the smooth nature of the flow to a viewer. For such purposes, the classic continuous methods are preferable over the combinatorial ones. Second, numerical algorithms require a larger number of parameters, which are often difficult to choose. In this example, the continuous version misses some saddle connectors, since a certain maximal number of integration steps had to be chosen for the extraction algorithm [21]. Of course, we could have changed that parameter and re-run the algorithm by Theisel et al. [21], but this still would not make it a proofably watertight case. Our combinatorial algorithm, on the other hand, captures all connectors by design. Hence, combinatorial methods are preferable over continuous ones if proofable correctness is the primary goal, e.g., if the extraction results are supposed to serve as an input for a further analysis.

5   Conclusions and Future Work

We presented a novel combinatorial algorithm to construct a weighted hierarchy of combinatorial gradient vector fields for 3D scalar data. The hierarchy represents the combinatorial flow for different levels of detail and implicitly defines the extremal structures. The weighting enables a distinction between spurious and dominant extremal structures. The hierarchy is efficiently represented by a sequence of augmenting paths. As seen in Table 1 the running time scales reasonable for common data sets. The memory consumption of Algorithm 1 is bounded. This allows for an analysis of large data sets. Our algorithm could allow for an extension of other methods that make use of a combinatorial gradient vector field such as topological smoothing [23] or tracking of critical points [17] to 3D.

References

1. Bauer, U., Lange, C., Wardetzky, M.: Optimal topological simplification of discrete functions on surfaces. CoRR **abs/1001.1269** (2010)
2. Cayley, A.: On contour and slope lines. Lond. Edinb. Dublin Phil. Mag. J. Sci. **18**, 264–268 (1859)
3. Chari, M.K.: On discrete Morse functions and combinatorial decompositions. Discrete Math. **217**(1-3), 101–113 (2000)
4. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. Discrete Comput. Geom. **28**(4), 511–533 (2002)
5. Forman, R.: Morse theory for cell complexes. Adv. Math. **134**, 90–145 (1998)
6. Gröller, M.E., Glaeser, G., Kastner, J.: Stagbeetle. http://www.cg.tuwien.ac.at/research/publications/2005/dataset-stagbeetle/
7. Gyulassy, A.: Combinatorial construction of Morse-Smale complexes for data analysis and visualization. Ph.D. thesis, University of California, Davis (2008)
8. Gyulassy, A., Bremer, P.T., Pascucci, V., Hamann, B.: Practical Considerations in Morse-Smale Complex Computation, chap. 6, pp. 67–78. Springer, Berlin (2009)
9. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge, U.K. (2002)
10. Joswig, M., Pfetsch, M.E.: Computing optimal Morse matchings. SIAM J. Discret. Math. **20**(1), 11–25 (2006)
11. King, H., Knudson, K., Mramor, N.: Generating discrete Morse functions from point data. Exp. Math. **14**(4), 435–444 (2005)
12. Lewiner, T.: Geometric discrete Morse complexes. Ph.D. thesis, Dept. of Mathematics, PUC-Rio (2005)
13. Lewiner, T., Lopes, H., Tavares, G.: Optimal discrete Morse functions for 2-manifolds. Comput. Geom. **26**(3), 221–233 (2003)
14. Maxwell, J.C.: On hills and dales. Lond. Edinb. Dublin Phil. Mag. J. Sci. **40**, 421–425 (1870)
15. Milnor, J.: Topology from the differentiable viewpoint. University Press, Virginia (1965)
16. Reininghaus, J., Günther, D., Hotz, I., Prohaska, S., Hege, H.C.: TADD: A computational framework for data analysis using discrete Morse theory. In: Mathematical Software – ICMS 2010, pp. 198–208. Springer, Berlin (2010)
17. Reininghaus, J., Kasten, J., Weinkauf, T., Hotz, I.: Combinatorial feature flow fields: Tracking critical points in discrete scalar fields. Tech. Rep. 11-02, Zuse Institute Berlin (2011)
18. Robins, V., Wood, P.J., Sheppard, A.P.: Theory and algorithms for constructing discrete morse complexes from grayscale digital images. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1646–1658 (2011). doi:10.1109/TPAMI.2011.95
19. Röttger, S.: The volume library. http://www9.informatik.uni-erlangen.de/External/vollib/
20. Stalling, D., Steinke, T.: Visualization of Vector Fields in Quantum Chemistry. Zuse Institute, Berlin (1996)
21. Theisel, H., Weinkauf, T., Hege, H.C., Seidel, H.P.: Saddle Connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In: Proceedings IEEE Visualization 2003, pp. 225–232. Seattle, U.S.A. (2003)
22. Weinkauf, T.: Extraction of topological structures in 2d and 3d vector fields. Ph.D. thesis, University Magdeburg and Zuse Institute Berlin (2008)
23. Weinkauf, T., Gingold, Y., Sorkine, O.: Topology-based smoothing of 2D scalar fields with C^1-continuity. Computer Graphics Forum (Proc. EuroVis) **29**(3), 1221–1230 (2010)
24. Zomorodian, A.: Computing and comprehending topology: Persistence and hierarchical Morse complexes. Ph.D. thesis, Urbana, Illinois (2001)

Computing Simply-Connected Cells in Three-Dimensional Morse-Smale Complexes

Attila Gyulassy and Valerio Pascucci

1 Introduction

Topology-based techniques can be used to extract features robustly from large and complex data. The Morse-Smale (MS) complex is especially well-suited in this context, because it encodes a large features space. Once computed, the cells of the MS complex identify regions of the data with uniform gradient flow properties: often, the problem of extracting features is reduced to selecting which cells of the complex to extract. For example, in recent combustion experiments, one hypothesis states that flame extinction occurs in regions of mixture fraction that can be described by simply-connected crystals of the MS complex. Testing this theory hinges on extracting valid cells; as we will show, previous techniques fall short of computing all cells of three-dimensional MS complexes.

The following contributions are made in this paper:

- We identify the challenges in extracting topologically valid cells from a discrete gradient vector field.
- We present an algorithm that computes the distinct cells of the MS complex connecting two critical points.
- We propose data structures to enable an efficient implementation of the algorithm.

A. Gyulassy (✉) · V. Pascucci
SCI Institute, University of Utah, UT, USA
e-mail: jediati@sci.utah.edu; pascucci@sci.utah.edu

2   Related Work

Topology-based techniques are well-studied in the context of scalar function analysis. Typically, a topological representation of the function is computed, then queried. The MS complex is a topological data structure that provides an abstract representation of the gradient flow behavior of a scalar field [1, 2]. Edelsbrunner et al. [3] presented the first algorithm for two-dimensional piecewise linear (PL) data, and Bremer et al. [4] improved this technique by following gradients more faithfully and describing a multi-resolution representation of the scalar field. For two-dimensional simply connected domains, there is a local property of orientation. Every cell of the complex can be computed by "walking" around the nodes and arcs of the complex. The interior of a quad can then be recovered by flood-filling from the boundary [4].

The problem becomes more difficult for three-dimensional domains, and various techniques have been proposed to handle this increased complexity. Edelsbrunner et al. [5] proposed an algorithm to compute the nodes, arcs, and quads of a three-dimensional MS complex for a PL function on a simplicial complex, however, a practical implementation was never devised due to the complexity of the algorithm. In particular, maintaining separation and ordering between ascending and descending manifolds proved challenging. The one-skeleton (nodes and arcs) of the MS complex was first computed successfully for volumetric data by Gyulassy et al. [6] by simplifying a dense "artificial" complex. Although the same authors later presented a more efficient approach to computing the MS complex by using a sweeping plane [7], data size and computational overhead still proved to be a limiting factor, despite not maintaining two- and three-dimensional manifolds of the critical point.

Recently, several techniques have been presented to compute components of the three-dimensional MS complex based on Forman's discrete Morse theory [8]. This is an attractive alternative to PL Morse theory since it simplifies the search for higher dimensional manifolds by discretizing the flow operator, and the ascending and descending manifolds are naturally separated by dimension. In [9–11] the authors presented techniques to compute a discrete gradient vector field. The 1-skeleton of the MS complex is simply recovered from a discrete gradient field as the critical cells and gradient paths connecting them. The descending and ascending manifolds are recovered as the sum of the cells reachable from a critical cell with the flow operator, and its inverse, respectively. While these approaches are practical and easy to implement, they fall short of identifying uniquely cells of the complex: in particular, they can not distinguish between two distinct cells having the same origin and destination. Currently, no algorithm is able to distinguish in practice between multiple cells connecting the same two critical points for three-dimensional MS complexes.

3 Background

Scalar valued volumetric data is most often available as discrete samples at the vertices of an underlying mesh. Morse theory has been well-studied in the context of smooth scalar functions, and has been adapted for such discrete domains. In the following, we briefly introduce Morse theory, to gain intuition of certain properties we want to maintain in the discrete case. Next, we review fundamental concepts from discrete Morse theory, and describe the discrete analogue to smooth Morse theory.

3.1 Morse Functions and the MS Complex

Let f be real-valued smooth map $f : \mathbb{M} \to \mathbb{R}$ defined over a compact d-manifold $\mathbb{M}$. A point $p \in \mathbb{M}$ is critical when $\nabla f(p) = \mathbf{0}$, i.e. the gradient is zero, and is non-degenerate when its Hessian (matrix of second partial derivatives) is non-singular. The function f is a *Morse function* if all its critical points are non-degenerate and no two critical points have the same function value. The *Morse Lemma* states that there exist local coordinates around p such that f has the following *standard form*: $f_p = \pm x_1^2 \pm x_2^2 \cdots \pm x_n^2$. The number of minus signs in this equation gives the *index* of critical point p. In three-dimensional functions, minima are index-0, 1-saddles are index-1, 2-saddles are index-2, and maxima are index-3.

An integral line in f is a maximal path in $\mathbb{M}$ whose tangent vectors agree with the gradient of f at every point along the path. Each integral line has an origin and destination at critical points of f. The *ascending* and *descending* manifold of a critical point p are obtained as clusters of integral lines having p as their common origin and destination, respectively. The descending manifolds of f form a cell complex that partitions $\mathbb{M}$; this partition is called the *Morse* complex. Similarly, the ascending manifolds also partition $\mathbb{M}$ into a cell complex. In a *Morse-Smale function*, integral lines only connect critical points with different indices. For volumetric domains, an index-i critical point has an i-dimensional descending manifold and a $(3 - i)$-dimensional ascending manifold. The intersection of the ascending and descending manifolds of a Morse-Smale function forms the *Morse-Smale (MS) complex*. Figure 1 illustrates the ascending and descending manifolds as well as the full MS complex for a two-dimensional domain.

The cells of this complex for functions defined on three-dimensional domains are *nodes*, *arcs*, *quads*, and *crystals*, of dimensions 0, 1, 2, and 3, respectively. The cells of the complex are well-structured. Figure 2 illustrates the various cells as well as the boundary relations between them. Each cell contains the integral lines that share a common origin and destination, along with the property that one line can be continuously deformed into another in the interior of the cell, i.e. the interior of a d-cell is a topological ball.

Fig. 1 A two-dimensional
Morse-Smale function is
decomposed into ascending
(**a**) and descending (**b**)
manifolds. The cells of the
Morse-Smale complex are
formed by the intersection of
these (**c**)
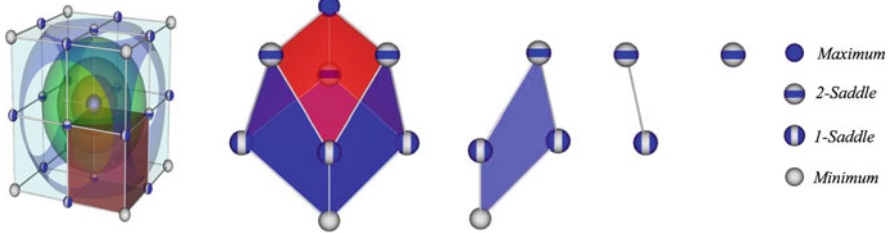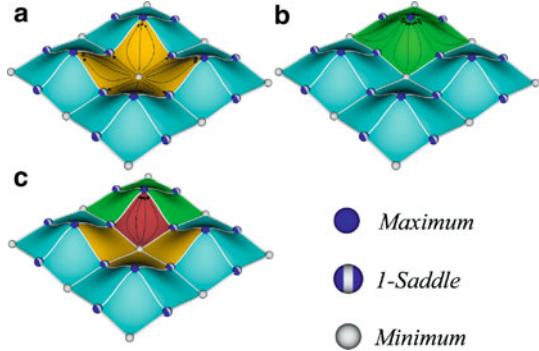
Fig. 2 The the MS complex for a three dimensional domain (*left*), with one crystal highlighted.
The cells (*middle*) of decreasing dimension from *left* to *right*: a *crystal* is a 3-cell that has one
minimum, one maximum, and an alternating ring of 1-saddles and 2-saddles. The facets of a crystal
are *quads*, quadrangular 2-cells. The facets of quads are the *arcs*, integral lines connecting critical
points with indices differing by one. The facets of arcs are *nodes*, critical points of the function

3.2 Discrete Morse Theory

Practical algorithms for computing MS complexes for volumetric data rely on
discretizing the domain. The following basic definitions from discrete Morse theory
are due to Forman [8]. A d-cell is a topological space that is homeomorphic to a
d-ball $B^d = \{x \in \mathbb{E}^d : |x| \leq 1\}$. For cells α and β, $\alpha < \beta$ means that α is a *face*
of β and β is a *co-face* of α, i.e., the vertices of α are a proper subset of the vertices
of β. If $dim(\alpha) = dim(\beta) - 1$, we say α is a *facet* of β, and β is a *co-facet* of α.
A cell α with dimension d is denoted $\alpha^{(d)}$. The *boundary operator* ∂ maps a cell to
its facets, and induces an orientation on the facets. The *face operator* $\bar{\partial}$ maps a cell
to its faces.

Let K be a regular complex that is a mesh representation of $\mathbb{M}$, the underlying
space. A function $f : K \rightarrow \mathbb{R}$ that assigns scalar values to every cell of K is a
discrete Morse function if for every $\alpha^{(d)} \in K$, its number of co-faces $|\{\beta^{(d+1)} >
\alpha | f(\beta) \leq f(\alpha)\}| \leq 1$, and its number of faces $|\{\gamma^{(d-1)} < \alpha | f(\gamma) \geq f(\alpha)\}| \leq 1$.
A cell $\alpha^{(d)}$ is critical if its number of co-faces $|\{\beta^{(d+1)} > \alpha | f(\beta) \leq f(\alpha)\}| = 0$
and its number of faces $|\{\gamma^{(d-1)} < \alpha | f(\gamma) \geq f(\alpha)\}| = 0$. To distinguish between

cells of the MS complex and critical cells of the mesh, we refer to critical cells as critical points.

A *vector* in the discrete sense is a pair of cells $\{\alpha^{(d)} < \beta^{(d+1)}\}$, where we say that an arrow points from $\alpha^{(d)}$ to $\beta^{(d+1)}$. Intuitively, this vector simulates a direction of flow. A *discrete vector field* V on K is a collection of pairs $\{\alpha^{(d)} < \beta^{(d+1)}\}$ of cells of K such that each cell is in at most one pair of V. Given a discrete vector field V on K, a V-*path* is a sequence of cells

$$\alpha_0^{(d)}, \beta_0^{(d+1)}, \alpha_1^{(d)}, \beta_1^{(d+1)}, \alpha_2^{(d)}, \dots, \beta_r^{(d+1)}, \alpha_{r+1}^{(d)}$$

such that for each $i = 0,\dots$, r, the pair $\{\alpha^{(d)} < \beta^{(d+1)}\} \in V$, and $\{\beta_i^{(d+1)} > \alpha_{i+1}^{(d)} \neq \alpha_i^{(d)}\}$. A V-path is the discrete equivalent of a streamline in a smooth vector field. A discrete vector field in which all V-paths are monotonic and do not contain any loops is a *discrete gradient field*. Note that in Forman's discrete Morse theory, gradient vectors point in the direction of decreasing function value. Cells that are not part of a gradient vector are critical with index of criticality equal to the dimension of the cell.

The descending manifold of a critical cell α is the sum of V-paths starting at α, and the ascending manifold is the sum of V-paths ending at α. Both the ascending and descending manifolds are sets of cells, and we can define the discrete Morse-Smale complex as the cells of the intersections of these manifolds. We say a cell of the MS complex is valid if it is a topological ball, and its boundary is a topological sphere made up of lower dimensional cells of the complex.

4 Challenges

Two main challenges must be overcome to extract valid cells of the MS complex in the discrete setting for volumetric domains: first, multiple cells of the complex may share critical points p and q as their origin and destination; second, the ascending and descending manifolds of an index-i critical point p, are not necessarily topological balls of dimension $(3-i)$ and i, respectively. Figure 3 illustrates the first problem. Here, two quads share the same minimum and maximum. These cells are merged together by techniques that only look at the intersection of ascending and descending manifolds to obtain the cells of the MS complex. In two-dimensions, multiple quads may share the same minimum and maximum; in three dimensions, multiple quads may share the same saddle and extremum, and multiple crystals may share the same minimum and maximum.

The second challenge in computing valid cells of the three-dimensional MS complex occurs because gradient paths can merge and split in the discrete setting. In particular, in computing the ascending 2-manifolds of 1-saddles and the descending 2-manifolds of 2-saddles, a naive approach of gathering cells in V-paths can create a non-manifold surface, as illustrated in Fig. 4.

a	4	5	6	9	8
	3	7	8	10	7
	2	5	1	8	6
	1	4	3	7	5
	0	1	2	3	4
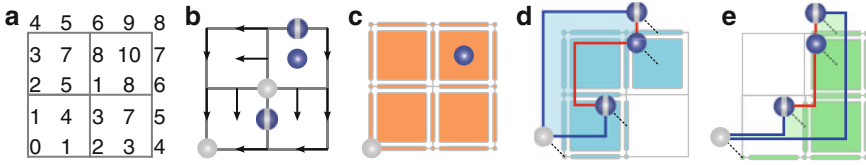
Fig. 3 We consider a two-dimensional discrete Morse function (**a**) and its discrete gradient field (**b**). The ascending manifold of the bottom left minimum is identical to the descending manifold of the maximum. These cells are highlighted in (**c**). The intersection of these manifolds is every cell except for the middle vertex. However, as illustrated in (**d**) and (**e**), there are two distinct quads of the two-dimensional MS complex in this intersection
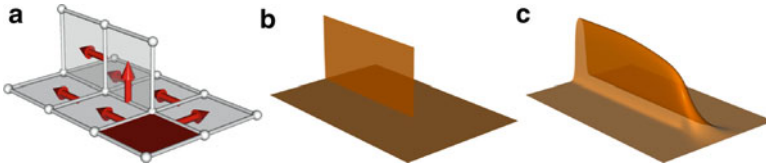
Fig. 4 Gradient vectors are shown for cells contained in V-paths originating at the 2-saddle (*red quad*) in a volumetric discrete gradient field. A simple union of these cells form a non-manifold surface (**b**). We can resolve this (**c**) by respecting the multiplicity of a cell in the descending manifold

5   Algorithm

One fundamental operation in using a MS complex in data analysis is to find the cells of the complex connecting an index-i critical point p with an index-j critical point q. Assume $i > j$. The dimension of the desired cells is $(j - i)$. Furthermore, each $(j - i)$-cell should reference the $((j - i) - 1)$-dimensional cells in its boundary. The algorithm we present in this section computes a representation of both the interior and the boundary cells. We divide the algorithm by cases, based on dimension.

5.1   Nodes: $j - i = 0$

When $j = i$, critical points p and q are not connected by any cells of the complex unless $p = q$. In that case, the interior of the node is identically p, and its boundary is empty.

5.2   Arcs: $j - i = 1$

When the indices of p and q differ by one, they are either connected by arcs of the complex or are not connected. Each V-path in the discrete gradient from p to q is

the interior of one arc of the complex, and its boundary is $\{p, q\}$. The V-paths can be computed using depth first search in the gradient vector field, as done in [9, 11].

5.3   Quads: $j - i = 2$

When the indices of p and q differ by two, they may be connected by quads of the MS complex. Either p or q must be a saddle with the other being an extremum. We compute quads of the three-dimensional MS complex by first explicitly computing ascending 2-manifolds from 1-saddles and descending 2-manifolds from 2-saddles. In each case, each cell of the input mesh is represented as a single vertex, and the facet/cofacet relation between cells is an edge. Without loss of generality, we will focus on computing descending 2-manifolds from 2-saddles. Once the 2-manifold is computed, we compute the interior of quads connecting p and q by flood-filling from each instance of the extremum found on the boundary of the 2-manifold. The boundary of the quads can be found by traversing the 2-manifold. This is described in detail in the next subsections.

5.3.1   Descending 2-Manifolds

Intuitively, we build a mesh representation of the descending 2-manifold by extending a pre-existing disk D, growing from its boundary. A circular list L of the cells around the boundary of D is maintained. D is extended locally based on the discrete gradient, updating L in the process. Finally, the boundary is zipped and finalized. The resulting mesh simulates a "flood fill" of the descending manifold that maintains disjointness where the mesh folds in on itself. In the following, we construct D and L with references to cells from the underlying mesh, since a single cell may be represented multiple times in the disk or its boundary. In each stage, we maintain two invariants: first, L is a circular list alternating references to 0- and 1-cells; second, D represents a simply-connected 2-manifold. Figure 5 illustrates each of the following stages of the algorithm.

Input: A 2-saddle p (a critical 2-cell), the underlying mesh K with a discrete scalar function, and the discrete gradient vector field V.

Initial conditions: Let r_p be a reference to p. Insert r_p and references to all the faces of p in K, i.e., all r_σ such that $\sigma \in \bar{\partial}(\alpha)$, into an empty surface mesh D, and add edges to this mesh to represent facet relations found in K, i.e., the edge (r_σ, r_α) is in D iff $\sigma \in \partial(\alpha)$. Then, r_p is classified as interior, and references to its faces are classified as unfinalized boundary. We pick an orientation for p, and initialize a circular list L with the references to its faces in D, maintaining the orientation.

Extend disk: We extend the disk locally by attaching a 2-cell to the boundary of D. Formally, select $r_\alpha \in L$ such that it references a 1-cell $\alpha \in K$ that is the tail of a
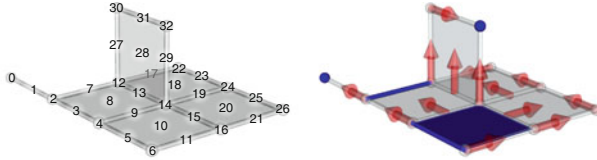
Input: The underlying mesh K is shown on the left, with each cell uniquely labeled. The gradient field V is shown on the right. Critical cells are marked with blue. We wish to compute the descending manifold of cell 10.
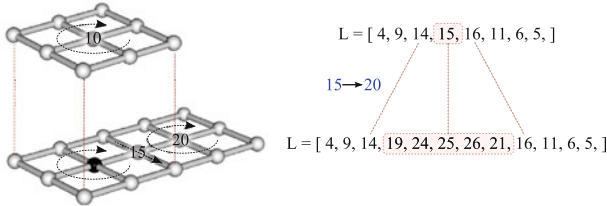
Initial + Expand Disk: The disk D and list L are initialized (top) with the local neighborhood of 10. Black nodes indicate references classified as interior to D. The disk is expanded with the gradient vector $\{15, 20\}$, and D and L are updated (bottom).
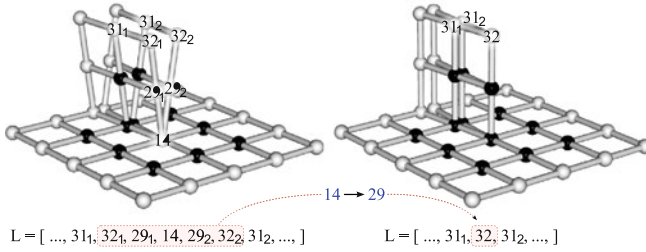
Zip Creases: The disk D and after all expanding (left). The sequence $[31_1, 32_1, 29_1, 14, 29_2, 32_2, 31_2]$ is zipped with the gradient vector $\{14, 29\}$. The subscripts denote distinct elements of L referencing the same cells in K. The resulting disk and boundary (right) cannot be further zipped.
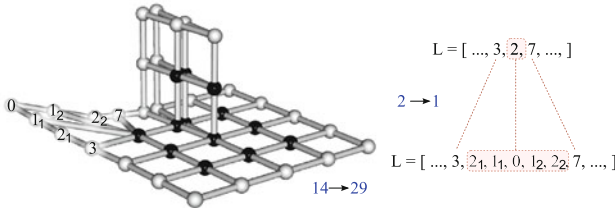
Finalize Boundary: The boundary is first extended, maintaining disjointness of gradient paths. Node 2 is found to be a corner and extended along the gradient vector $\{2, 1\}$. The result of this extension is the final state for D and L.

Fig. 5 We show the various stages of the algorithm. The input is mesh is K, however we only show the sub-mesh that will participate in the descending 2-manifold of a critical 2-cell

gradient vector in V. Let μ and ν be the two vertices of α in K. The circular list L is guaranteed to contain the references to μ and ν, r_μ and r_ν. In fact, r_μ and r_ν occur right before and after r_α in L, respectively, and the ordered list $[r_\mu, r_\alpha, r_\nu]$ implies a direction. Let β be the 2-cell at the head of the gradient vector starting at α, and let $\bar{\delta}(\beta) = [r_\mu, r_\alpha, r_\nu, r_{\gamma 1}, r_{\gamma 2}, r_{\gamma 3}, r_{\gamma 4}, r_{\gamma 5}]$ be the ordered circular list of the faces on the boundary of β. We extend D by inserting references to $\beta \cup \bar{\delta}(\beta) \setminus \{\alpha, \mu, \nu\}$, and adding edges to D based on the facet relation. Mark the newly inserted references to faces of β as unfinalized boundary and r_α and r_β as interior.

The circular list L is updated with the newly inserted references as well, with the opposite orientation as implied by $[r_\mu, r_\alpha, r_\nu]$. More formally, let $\bar{\delta}(v_\beta) \setminus [r_\mu, r_\alpha, r_\nu] = [r_{\gamma 1}, r_{\gamma 2}, r_{\gamma 3}, r_{\gamma 4}, r_{\gamma 5}]$ be the circular list of face references of β, broken by the removal of $[r_\mu, r_\alpha, r_\nu]$. Order $\bar{\delta}(v_\beta) \setminus [r_\mu, r_\alpha, r_\nu]$ with opposite orientation as implied by $[r_\mu, r_\alpha, r_\nu]$, such that the first element of this list is the next reference after r_μ in $\bar{\delta}(\beta)$. Then, the local update to the circular list L is realized by:

$$\ldots, \; r_\mu, \; r_\alpha, \; r_\nu, \; \cdots \rightarrow \ldots, \; r_\mu, \; \bar{\delta}(v_\beta) \setminus [r_\mu, r_\alpha, r_\nu], \; r_\nu, \ldots$$

$$= \ldots, \; r_\mu, \; r_{\gamma 5}, \; r_{\gamma 4}, \; r_{\gamma 3}, \; r_{\gamma 2}, \; r_{\gamma 1}, \; r_\nu, \ldots$$

Zipping the disk: The boundary of the disk is represented by the circular list L. After extending the disk, the interior vertices of D are references to the 1- and 2-cells in V-paths whose destination is p. We use the gradient defined on 0- and 1-cells to finish the "flood fill" and zip up the creases in the disk, and to mark these cells as interior to the disk. Note that this is not a purely geometric operation, since a crease may in fact be a topological feature. Let r_α in L reference a 0-cell α in K, and α is the tail of a gradient vector in V. Since L is a list of alternating 0- and 1-cell references, the 2-neighborhood of r_α in L can be written as the ordered list $[r_\mu, r_\beta, r_\alpha, r_\gamma, r_\nu]$, where r_μ and r_ν reference 0-cells in K, and r_β and r_γ reference 1-cells. The boundary can be zipped up iff: $\mu = \nu$, $\beta = \gamma$, $\{\alpha < \beta\}$ is a gradient vector in V, and references to all cofacets excluding β and γ of α in a 1-neighborhood of r_α in D are marked interior to D. The zip operation merges r_μ and r_ν, and r_β and r_γ in D, as well as the edge (r_μ, r_β) with (r_ν, r_γ) and the edge (r_β, r_α) with $(_\gamma, r_\alpha)$. The merged r_μ and r_α are marked interior to D. The local update to L is realized by:

$$\ldots, \; r_\mu, \; r_\beta, \; r_\alpha, \; r_\gamma, \; r_\nu, \; \cdots \rightarrow \ldots, \; r_\mu, \; \ldots$$

Finalizing boundaries: After zipping up creases, L contains all the references classified as unfinalized boundary in D. In the MS complex, the boundary of a descending 2-manifold is a ring of descending 1-manifolds from 1-saddles. However, L only contains the boundary references that are faces of 2-cells referenced in D: the boundary of D may in fact extend past this in the case where two 1-manifolds merge. Therefore, the first step in resolving the boundaries is to extend "corners". Let $[r_\beta, r_\alpha, r_\gamma]$ be an ordered list from L, and r_α reference a 0-cell in K. We extend

the boundary iff α is the tail of a gradient vector $\{\alpha < \sigma\}$ in V and both β and γ are distinct from σ. Let μ be the 0-cell at the other end of σ from α. We extend the boundary with the following rule:

$$\ldots, r_\beta, r_\alpha, r_\gamma, \cdots \rightarrow \ldots, r_\beta, r_{\alpha 1}, r_{\sigma 1}, r_\mu, r_{\sigma 2}, r_{\alpha 2}, r_\gamma, \ldots$$

The subscripts indicate distinct instances that reference the same cells in K. The disk D is also extended, with the new references inserted. Edges are added between each new boundary reference. Finally, let r_μ be an interior node edge-connected to r_β, and r_v the interior node edge-connected to r_γ. The existence of these is guaranteed since D represents a disk. Add edges $(r_{\sigma 1}, r_\mu)$ and $(r_{\sigma 2}, r_v)$ to D. Intuitively, this extends the 2-cells containing r_α, stretching the disk along the gradient vector.

The boundary is finalized by tracing paths from boundary 1-saddles to boundary minima, marking each reference passed. Each reference in the finalized boundary is part of exactly one descending 1-manifold.

Discussion: The result of the above algorithm is a graph of references D and a list of 0- and 1-cells around its boundary, L. We can convert this into an actual surface S_D: each item in D references an i-cell in K, and can be converted into an i-cell in S_D. We create a cell complex that represents the surface by attaching faces based on the edges in D, as shown in Fig. 6a. Indeed there is a 1-to-1 mapping between references in D and cells in S_D. To avoid additional notation, we call cells in S_D by their reference names in D.

Each of the stages of the algorithm maintains both of the invariants stated above. The initial condition satisfies both invariants trivially: the oriented boundary of a 2-cell is an alternating list of 0- and 1-cells; the 2-cell and its faces form a disk. Extending the disk replaces a 1-cell on the boundary with a sequence of 1-, 0-, 1-, 0-, and 1-cells, maintaining the first invariant. Each zip step removes a contiguous even length sequence from the boundary, and each corner extension step in finalizing the boundary adds a 1-, 0-, 1-, 0-cell sequence right after a 0-cell; both these steps maintain the first invariant. None of the steps introduce or destroy connected components of the boundary of D. Furthermore, points newly interior to D after one step only create new edges connecting to boundary references. Therefore, D remains a disk throughout the algorithm.
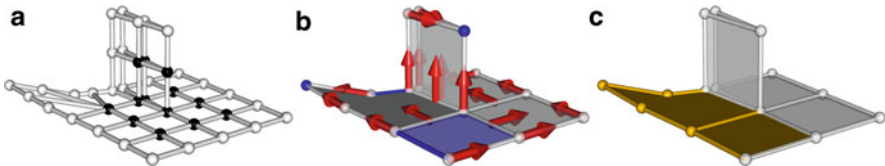
Fig. 6 The reference mesh D from Fig. 5 is converted into the surface S_D (**a**). A gradient field V_D is inherited from V on S_D (**b**). Tracing the ascending manifold of 0 (**c**) by summing V-paths of V_D gives the interior of one quad

5.3.2 Quads

Given a descending 2-manifold from a 2-saddle, represented as the disk surface S_D, computation of quads is easily achieved. First, we define a discrete gradient field V_D as the following: $\{r_\alpha < r_\beta\}$ is a gradient vector in V_D iff $r_\alpha, r_\beta \in S_D$ and $\{\alpha < \beta\}$ is a gradient vector in V. Figure 6b shows V_D computed for the example in Fig. 5. V_D is loop free, since every V-path in V_D references cells in a V-path in V, and hence, V_D is a discrete gradient field.

Quads are recovered from the descending 2-manifold by tracing ascending gradient paths on V_D from every instance of a minimum in S_D. This ascending manifold restricted to the descending 2-manifold gives the interior of a quad. The two saddle-extremum arcs of a quad are the paths along the boundary of S_D connecting a minimum to two 1-saddles, and ascending paths from those saddles form the two 1-saddle-2-saddle connections.

5.3.3 Ascending 2-Manifolds

We can compute ascending 2-manifolds by considering the dual complex of K, with every gradient vector in V reversed. In this case, a 1-saddle in K becomes a 2-saddle in the dual, and the ascending manifold of the 1-saddle in K is identical to the descending manifold of the 2-saddle in the dual.

5.4 Crystals: $j - i = 3$

The following algorithm computes the boundary and interior of each distinct crystal connecting a minimum and maximum in the MS complex. Unlike with quads, we take an indirect approach in computing crystals, as the data structures and rules for extending a sphere surface become elaborate. First, we compute the 2-manifolds of all saddles connected by arcs of the MS complex to the minimum or maximum, and represent them in an oriented quad-ring data structure that encodes the relative position of each quad. Then, a walk on this structure recovers the boundary of a single crystal. We repeat this walk for all crystals. The interior of each crystal is then filled in.

Input: A minimum mn, a maximum mx, the underlying mesh K, and the discrete gradient vector field V.

Oriented quad-ring: Let s be a 2-saddle; it is a critical 2-cell that is the boundary of exactly two 3-cells that are the first steps in V-paths a_1 and a_2 leading to maxima mx_1 and mx_2, respectively. In computing 2-manifolds, we assign an orientation to the saddle, and using a right hand rule, we can write the ordered list $[a_1, s, a_2]$. Here, mx_1 is "below" and mx_2 is "above" the oriented descending 2-manifold of s.

Furthermore, a quad inherits its orientation from an oriented 2-manifold, and we create a circular list with similar orientation that contains the cells in the quad's four boundary V-paths. Finally, quads are themselves ordered in a circular list around s with the same orientation. The oriented quad-ring data structure is composed of the list $[a_1, s, a_2]$ along with the circular list of oriented quads, each of which has an oriented circular list representing its boundary. Figure 7 illustrates this data structure. Given a saddle with orientation, we compute this data structure from its oriented 2-manifold: a walk around the boundary of the 2-manifold computes the circular list of quads; a walk around the boundary of each quad computes the oriented boundary list; and V-paths are traced in V using a right hand rule along with the orientation of s to recover a_1 and a_2.

Computing crystal boundaries: Given mn and mx, let A be the set of arcs of the MS complex that have one as an endpoint. For each arc $a_i \in A$, pick the saddle s opposite the extremum. If the extremum is a maximum, compute the 2-manifold with orientation induced from a right hand rule going from s to the extremum along a_i, and compute its oriented quad-ring. If the extremum is a minimum, compute the 2-manifold with orientation induced from a right hand rule going from the minimum to s along a_i, and compute its oriented quad-ring. Intuitively, a crystal will be found "below" an ascending 2-manifold, and "above" a descending 2-manifold. We can discard any quad that does not have either mn or mx in its boundary, since these cannot bound crystals between the two critical points.

We begin computing a crystal by selecting at random one oriented quad q_i in an oriented quad ring from a 1-saddle s_1. The arc a_1 below s_1 in the oriented quad ring data structure leads to mn, and mx is present opposite of s_1 in the boundary list of q_i. We compute an arc a_2 of the complex by walking from mx until the fist 2-saddle s_2, and compute the 1-saddle-2-saddle arc a_{12} by continuing that walk until s_1. Since saddle-extremum arcs are unique, we can find the oriented quad-ring of s_2 having a_2 above. Within that quad-ring, we can find the quad q_2 such that a walk around the boundary from s_2 first traces a_{12} and then a_1. Then q_2 is the next quad in our walk, and we repeat this same procedure, alternating between quads in descending 2-manifolds and ascending 2-manifolds, until we return to q_1. Figure 7 illustrates this process. The set of quads traversed form the boundary of one crystal in the MS complex. After the boundary of one crystal is found, the quads are removed from the oriented quad-ring data structures. The process of finding crystal boundaries is repeated until all candidate quads have been exhausted.

Computing crystal interiors: The boundary of a crystal is an alternating ring of quads. The boundaries of these quads are glued together to form a watertight shell, a topological sphere. Since the quads are oriented, we can begin a flood-fill of the interior by going down from a cell in a 1-saddle, 2-saddle, maximum, 2-saddle quad on the boundary, with respect to the right hand rule.
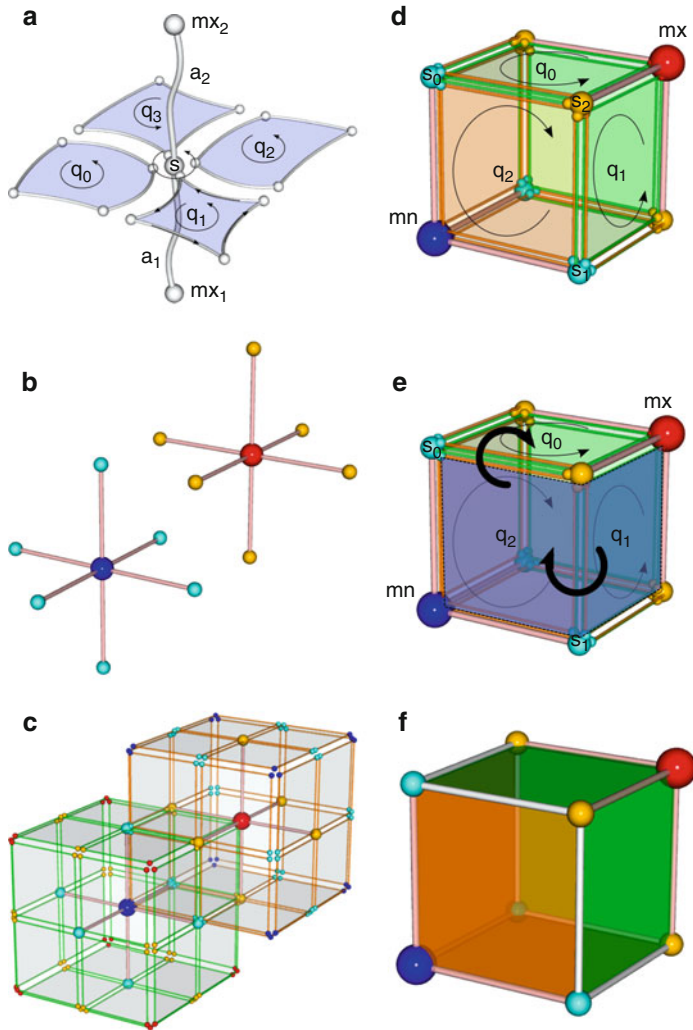
Fig. 7 The quad-ring data structure (**a**) maintains orientation of quads as well as which extremum, mx_1 or mx_2 is above or below the quads. In the first stage of finding crystals between a minimum (*blue*) and maximum (*red*), shown in (**b**), all arcs (*pink*) connecting 1-saddles (*teal*) to the minimum and 2-saddles (*yellow*) to the maximum are identified. Next, an oriented quad-ring is computed for each saddle using the induced orientation from the arc (**c**). After discarding all non-candidate quads (**d**), we start building the boundary of a crystal. We pick q_1 first: walking along the boundary from mx, we find s_2. We search for a quad in the oriented quad-ring (oriented by the arc (mx, s_2)) that contains the arc (s_2, s_1) with the same orientation as in q_1: and end up at q_2. We continue walking across 1-saddle-2-saddle arcs in this manner (**e**). The final crystal with glued boundaries is shown in (**f**)

6   Discussion

The algorithm we presented computes the distinct cells of the MS complex between any two critical points. During the computation, we only rely on basic data structures. In computation of quads, we used a circular list to store the boundary of a surface mesh representing the descending 2-manifold. Orientation can be achieved through ordering of the faces of a cell, and techniques are well-known for maintaining orientation in surface meshes. In computation of crystals, testing the equality of two arcs of the MS complex is simply implemented as a list comparison.

Although computation of the crystals between a minimum and maximum is potentially computationally expensive, it is expected that in most cases there is only one crystal connecting the extrema, and the search for candidates in the walk around 1-saddle-2-saddle is reduced to examining the one possible candidate. Indeed, it is not necessary to compute the full quad-ring since we only are concerned with quads that terminate at our input extrema: by flood-filling and marking the 3-manifolds of the extrema we wish to query in K, we can construct the 2-manifolds restricted to these marked cells.

7   Conclusions

We identified the challenges in computing cells of the MS complex for volumetric domains, and presented an algorithm to overcome these problems. Each step of the algorithm is combinatorial, and only relies on maintaining orientations on surfaces, avoiding the complexity of ordering surfaces in three dimensions. The framework provided by discrete Morse theory is at the root of the simplicity of the algorithm; the fact that ascending and descending manifolds are naturally separated by dimension in this setting makes this algorithm realizable.

References

1. Smale, S.: On gradient dynamical systems. Ann. Math. **74**, 199–206 (1961)
2. Smale, S.: Generalized Poincaré's conjecture in dimensions greater than four. Ann. Math. **74**, 391–406 (1961)
3. Edelsbrunner, H., Harer, J., Zomorodian., A.: Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. Discrete Comput. Geom. **30**(1), 87–107 (2003)
4. Bremer, P.T., Edelsbrunner, H., Hamann, B., Pascucci, V.: A topological hierarchy for functions on triangulated surfaces. IEEE Trans. Visual. Comput. Graph. **10**(4), 385–396 (2004)
5. Edelsbrunner, H., Harer, J., Natarajan, V., Pascucci., V.: Morse-Smale complexes for piecewise linear 3-manifolds. In: Proceedings of the 19th Annual Symposium on Computational Geometry, pp. 361–370 (2003)
6. Gyulassy, A., Natarajan, V., Pascucci, V., Bremer, P.T., Hamann, B.: Topology-based simplification for feature extraction from 3d scalar fields. In: Proceedings of IEEE Conference on Visualization, pp. 535–542 (2005)

7. Gyulassy, A., Natarajan, V., Pascucci, V., Bremer, P.T., Hamann, B.: A topological approach to simplification of three-dimensional scalar functions. IEEE Trans. Visual. Comput. Graph. **12**(4), 474–484 (2006)
8. Forman, R.: A users guide to discrete Morse theory. In: Proceedings of the 2001 International Conference on Formal Power Series and Algebraic Combinatorics, A special volume of Advances in Applied Mathematics, p. 48 (2001)
9. Lewiner, T., Lopes, H., Tavares, G.: Applications of Forman's discrete Morse theory to topology visualization and mesh compression. IEEE Trans. Visual. Comput. Graph. **10**(5), 499–508 (2004)
10. King, H., Knudson, K., Mramor, N.: Generating discrete Morse functions from point data. Exp. Math. **14**(4), 435–444 (2005)
11. Gyulassy, A., Bremer, P.T., Hamann, B., Pascucci, V.: A practical approach to Morse-Smale complex computation: Scalability and generality. IEEE Trans. Visual. Comput. Graph. **14**(6), 1619–1626 (2008)

Combinatorial Vector Field Topology in Three Dimensions

Wieland Reich, Dominic Schneider, Christian Heine, Alexander Wiebel, Guoning Chen, and Gerik Scheuermann

1 Introduction

Topology-based methods are of increasing importance in the analysis and visualization of datasets from a wide variety of scientific domains such as biology, physics, engineering, and medicine. Especially in the context of vector fields great research efforts have been undertaken to segment the domains of the available data into meaningful regions. In particular, steady vector field topology tries to find regions in which streamlines exhibit similar behavior. These regions can be used for further processing and analysis of the vector field itself or to simplify the visualization. The latter is usually achieved by drawing only the region's borders, the so called separatrices. This produces less geometry and thus less visual clutter than illustrating all particulars of the field. Both of these advantages are relevant for two-dimensional vector fields, but become critical for three-dimensional vector fields where possible occlusion appears as additional problem.

Different methods for vector field topology in two as well as in three dimensions have been proposed in the past. The most recent advances come from the sub-field of combinatorial vector field topology. Unfortunately, up to now only techniques for two-dimensional fields have been presented in this context so far. In this paper

W. Reich (✉) · D. Schneider · C. Heine · G. Scheuermann
University of Leipzig, Leipzig, Germany
e-mail: reich@informatik.uni-leipzig.de; schneider@informatik.uni-leipzig.de;
heine@informatik.uni-leipzig.de; scheuermann@informatik.uni-leipzig.de

A. Wiebel
Max-Planck-Institut for Human Cognitive and Brian Sciences, Leipzig, Leipzig, Germany
e-mail: wiebel@cbs.mpg.de

G. Chen
University of Utah, UT, USA
e-mail: chengu@sci.utah.edu

we try to fill the gap of missing combinatorial vector field topology methods for three dimensions. We present two methods to process three-dimensional fluid flows, which both use graph algorithms to extract features from the underlying vector field. We will apply the methods to several synthetic data sets and one of them to a CFD-simulation of a gas furnace chamber. In the end, we provide several options for categorizing the invariant sets respective to the flow.

2   Related Work

In the history of flow visualization – a survey can be found in [22,29] – topological methods make their steady appearance. Helman and Hesselink [13] introduced them to the visualization community starting with extracting and classifying singularities also known as critical points. Many other topological structures beyond singularities have been used in visualization. Periodic orbits have been subject to visualizations by Theisel et al. [27] and Wischgoll and Scheuermann [30]. Peikert and Sadlo [21] improved the display of invariant manifolds for saddle points and periodic orbits. Displaying all such invariant manifolds at once leads to an occlusion problem. A solution is to only display their intersection curves, the so-called saddle connectors [26]. Tricoche et al. [28] proposed vector field simplification based on topological methods. In the recent past Morse theory has gained interest in the analysis of vecter field data. Zhang et al. [31] and Chen et al. [3] introduced Morse decomposition and Conley index theory to the visualization community. Edelsbrunner et al. [6] and Guylassy et al. [9] use Morse-Smale complexes to process the gradient field obtained from scalar data. Reininghaus and Hotz [23] process 2-D vector fields with a method based on Forman's work [7]. Unlike our method, it transforms the cells and simplices of lower dimension, i.e. edges and vertices, directly into a graph. In contrast, our work is the extension of Morse decompositions of flows on two-dimensional manifolds, which has been subject of the work of Chen et al. [3, 4], to three dimensions. An example visualization obtained with [3] is given in Fig. 1.

3   Vector Fields, Flows, and Morse Decompositions

Let $X' = F(X)$ be a differential equation defined on $\mathbb{R}^3$, then the associated flow is a continuous function $\Phi : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^3$, satisfying

$$\Phi(0, x) = x, \tag{1}$$

$$\Phi(t_1, \Phi(t_2, x)) = \Phi(t_1 + t_2, x). \tag{2}$$
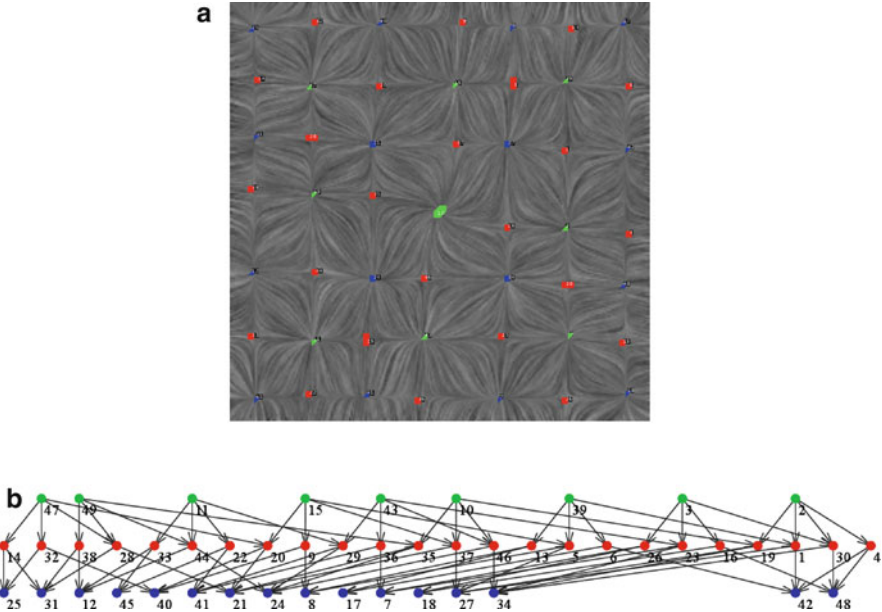
Fig. 1 A Morse decomposition of a planar field computed with the algorithm by Chen et al. [3] (**a**) A vector field visualized with a Line Integral Convolution. Extracted Morse sets are displayed as *green* (source), *red* (saddle), or *blue* (sink) (**b**) The resulting Morse connection graph. The nodes have the same color as the represented fixed points in the field above

We have

$$\frac{d}{dt}\Phi(t, x)_{|x_0} = F(x_0). \tag{3}$$

$S \subset \mathbb{R}^3$ is an *invariant set* if $\Phi(t, S) = S$ for all $t \in \mathbb{R}$. For example, the trajectory of any point $x \in \mathbb{R}^3$ is an invariant set.

A compact set $N \subset \mathbb{R}^3$ is called *isolating neighborhood* if the maximal invariant set S that is contained in N, lies in the interior of N. A set S is an *isolated invariant set* if there exists an isolating neighborhood N so that S is the maximal invariant set contained in N.

Hyperbolic fixed points and periodic orbits are examples of isolated invariant sets, but also the space of their connecting trajectories. Let $\varepsilon > 0$ be small, we define the *exit set* of an isolating neighborhood as

$$L = \{x \in \partial N \mid \Phi(t, x) \cap N = \emptyset\}, \ t \in \,]0, \varepsilon[\tag{4}$$

The so called index pair (N, L) will be of further interest in Sect. 6, when we classify isolated invariant sets.

The *alpha-* and *omega limit sets* of $x \in \mathbb{R}^3$ are

$$\alpha(x) = \bigcap_{t \in]-\infty, 0[} \overline{\Phi(t, x)} \tag{5}$$

$$\omega(x) = \bigcap_{t \in]0, \infty[} \overline{\Phi(t, x)} \tag{6}$$

where the overline denotes the closure of a set.

A *Morse decomposition* $\mathcal{M}$ of $X \subset \mathbb{R}^3$ is a finite collection of isolated invariant subsets of X, called *Morse sets* M:

$$\mathcal{M}(X) = \{M(p) \mid p \in \mathscr{P}\}, \tag{7}$$

such that if $x \in X$, then there exists $p, q \in \mathscr{P}$ such that $\alpha(x) \subset M(q)$ and $\omega(x) \subset M(p)$. In addition, there exist a partial order $>$ on $\mathscr{P}$ satisfying $q > p$ if there is an $x \in X$ such that $\alpha(x) \subset M(q)$ and $\omega(x) \subset M(p)$.

4 Geometry-Based Flow Combinatorialization

In this section we present an algorithm that computes regions with source, saddle, and sink-like behavior and the Morse connection graph between these regions, for a vector field defined on a simplicial 3-D mesh $\mathbf{T} = \bigcup T_i$.

We start by defining an equivalence relation between tetrahedra: $T_i \sim T_j$ if and only if there exists a sequence of connected tetrahedra $T_i, \ldots, T_j$ where all intermediate faces are non-transversal, i.e. flow through the face is not uni-directional. Each face can be easily checked by testing whether the flow vector at each of its three vertices produces the same sign when the inner product with the face normal is computed. For this procedure we use linear interpolation on the tetrahedral mesh. The resulting equivalence relation partitions the domain into polyhedral regions $\mathbf{R} = \bigcup R_i$, with all faces being transversal.

Then we construct a *flow graph*, which encodes a combinatorial description of the flow in the field. In this graph, each equivalence class of cells R_i is represented by a node u_i. If there exists a common face between two elements R_i and R_j of $\mathbf{R}$ and the flow points from R_i to R_j, we add an arc (u_i, u_j) in the graph.

We then compute the flow graph's strongly connected components using the popular algorithm by Tarjan [25]. A strongly connected component of a graph is a maximal subgraph in which for each pair of vertices u_i, u_j there exists a directed path from u_i to u_j. An example is given in Fig. 2. Strongly connected components describe regions of recurrent flow. As the strongly connected components induce an equivalence relation between graph nodes, we compute the quotient graph by adding one node v_i for each strongly connected component c_i and an arc from nodes v_i to v_j if there is an arc from some node of c_i to some node of c_j. It is trivial to show that
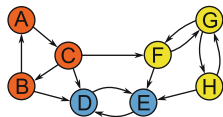
Fig. 2 A drawing of strongly connected components. Different colors describe the pairwise disjoint sets of nodes in this graph, in each existing a path from an arbitrarily chosen node to all others of the same component
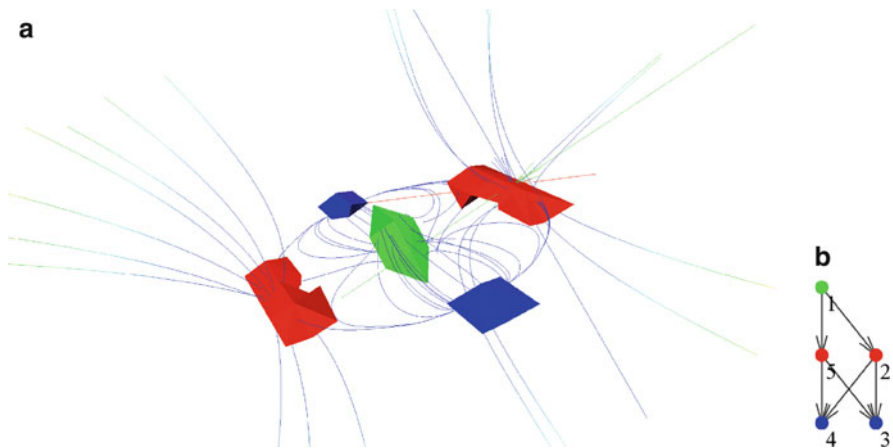
Fig. 3 A decomposition with the geometry-based algorithm of 3-D data containing two sinks (*blue*), a source (*green*), and two saddles (*red*) (**a**) Morse sets in a vector field (**b**) Morse connection graph

the quotient graph on the strongly connected components of a graph is an acyclic directed graph.

From the quotient graph we then remove all nodes that neither are sources or sinks nor contain a critical point with respect to vector field topology. The removed nodes represent trivial flow behavior, e.g. all entries in the Conley index are zero. The Conley index is a topological invariant discussed in Sect. 6. To preserve connectivity, we add an arc for each combination of the removed node's successors with its predecessors. The resulting graph's nodes represent regions which contain isolated invariant sets. A proof that the resulting graph, which we henceforth call Morse connection graph (MCG), encodes a Morse decomposition of the phase space was given in [1]. We classify each node, which represents a Morse set, according to Sect. 6. We then show the graph in an additional window using the algorithm by Gansner et al. [8] for graph layout. We restrict all sources and all sinks to be on one layer, respectively. Furthermore we remove all transitive arcs, i.e. arcs (v_i, v_j) for which there exists a path from v_i to v_j not using (v_i, v_j), as they complicate graph layout but do not improve perception ([5], Chap. 1).

This algorithm works well with regions in vector fields with gradient-like flow behavior (Fig. 3). For highly rotational fields, the purely geometry-based

method can separate only few regions. Theoretically, one could try to find a new tetrahedralization of the trajectory space, where all cells have transverse faces, i.e. unidirectional flow everywhere. This is a complicated task and instead we turn our attention to a streamline-based approach.

5   Streamline-Based Flow Combinatorialization

An improvement of the geometry-based Morse decomposition was explained in [4]: For each cell a set of streamlines is seeded uniformly across the cell's volume and integrated for a short time or length and the cells in which the streamlines end are recorded. A mathematical foundation, that this is leading to Morse decomposition of the phase space, is given in [17].

Having a tetrahedron T_1 in a mesh $\mathbf{T}$, then there exists a union of tetrahedra T_i, so that the image of T_1 under streamline integration lies completely in $\bigcup T_i$. We will refer to it as an outer approximation (Fig. 4). The resulting combinatorial multi-valued map $\mathscr{F} : \mathbf{T} \to \mathbf{T}$ is then encoded into the flow graph. We then proceed in the same way as geometry-based method, computing strongly connected components, quotient graph, and removing cells with trivial flow behavior.

Once the graph is created, we cannot influence the outcome anymore, so let us discuss the modalities of the integration. We used DoPri5 [10] and decided to integrate all tetrahedra by a fixed arc-length. The reason is simple, some cells in slowly moving regions will not have moved at all, while others may have reached the boundary. Furthermore, there are two possibilities to reconstruct the integration image, both using a sampling of the cell, so dense, that eventually no image cell will slip through the net. Using FTLE [11] or a similar predictor that just integrates the vertices to compute the stretching of the cell is not rigorous. Though, we could get
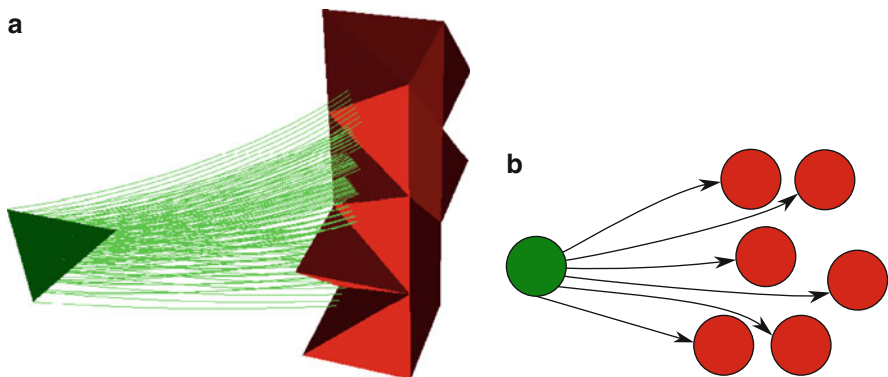
Fig. 4 Outer approximation (**a**) A union of cells (*red*) that encloses the image of the *green* cell under streamline integration (**b**) The multi-valued map is encoded into a graph

adequate results with that. A more rigorous, but computationally costly method is to integrate the uniformly distributed seeding points stepwise for small times and adaptively place new streamlines between them like in Hultquist's algorithm for stream surfaces [15]. Placing seeding points just on the boundary of the cell is an option, but does not save much of the computation time, since the whole volume must be reconstructed after the integration.

After computing the images of all seeding points, we collect their cell indices. Taking all of the neighbors of all computed image cells into the outer approximation is possible, but will lead to coarse results. Rigorous enclosing techniques are also discussed in [19].

6 Identification of Morse Sets

Finally, this algorithm is sensitive to the parameter of the integration length. Integrating for a too small arc-length will lead to many so called false positives, where highly spiraling flow does not differ from closed streamlines. In Fig. 5 we
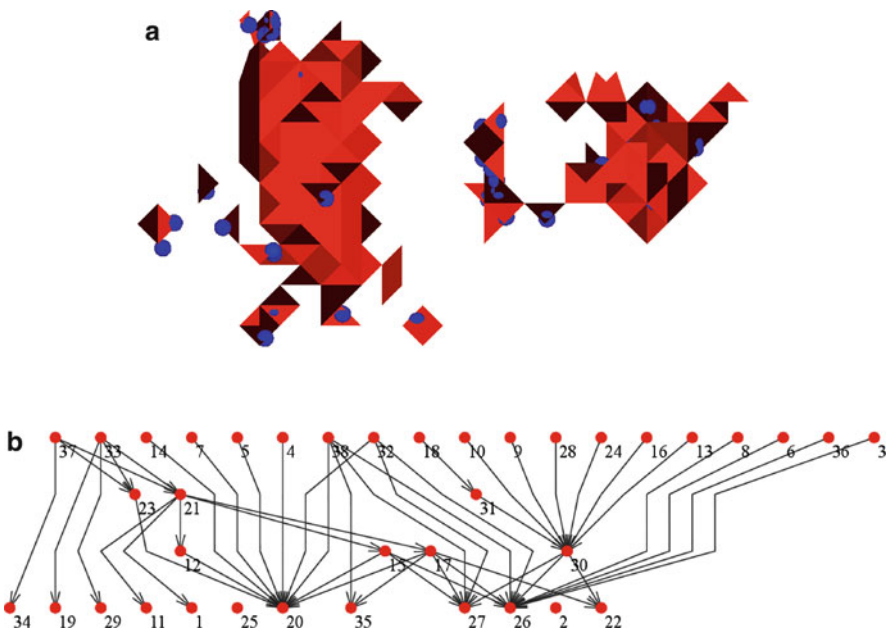
Fig. 5 A segment of a gas furnace chamber processed with the streamline-based Morse decomposition. The geometry-based algorithm was only able to extract one(!) Morse set in this highly turbulent flow (**a**) *blue*: spheres indicating saddle points, *red*: extracted Morse sets (**b**) As we expected, there are only saddle-like Morse sets in divergence free CFD-simulations, so we aligned the graph layout, that they are not displayed on a single line like in Fig. 1

applied the algorithm to a segment of a gas furnace chamber. We were able to find one closed streamline which was close together with a fixed point in the same Morse set. This reveals a weakness of our combinatorial algorithm: If the closed streamline is very small, then its outer approximation will topologically be a ball, not a torus. On the other hand, the question arises, whether a refinement is really necessary in all cases. The degree of simplification can be controlled by the arc-length-parameter and topological invariants are able to categorize a Morse set that consists of multiple fixed points and periodic orbits.

6.1 Classical Methods

For fixed points of first order, a commonly used method is the evaluation of the eigenvalues of the Jacobi matrix at the corresponding position. For a hyperbolic closed streamline, a Poincaré section plane can be used [14]. As we have no guarantee that these structures are always isolated from each other in our obtained Morse sets, we are not going to apply them.

6.2 Graph Analysis

A very simple, but coarse way to identify a Morse set is to enumerate the connected arcs before the node cancellation in the Morse connection graph. If all of the arcs are outgoing, it is a source. Analogous, if all arcs are incoming, it is a sink. If arcs of mixed types exist, it is a saddle-like Morse set. Eventually no more conclusions about the nature of the Morse set are possible.

6.3 Poincaré Index

The Poincaré index is a topological invariant, which is strongly related to the Conley index, but does not contain as much information. However, it has been successfully used to simplify vector field topology in two and three dimensions. It also can deal with fixed points of higher order [28].

6.4 Conley Index

A far more accurate topological invariant is the Conley index, which was introduced to the visualization community by Chen et al. [3], based on the comprehensive theoretical work by Mischaikow [18]. In [3,4], Morse decompositions and a method

of computing the Conley index in two dimensions are explained and a number of important indices were illustrated, i.e. those of hyperbolic fixed points and closed streamlines.

Let N be an isolating neighborhood and L its exit set as they were defined in Sect. 3.

Definition 1. The (homological) *Conley index* is defined as

$$CH_*(N) = H_*(N/L),$$

where $H_*(N, L)$ is the relative homology of the index pair (N, L).

The Conley index is able to predict the existence of invariant sets by checking whether inflow and outflow of its isolating neighborhood are in a nontrivial relation. It is a generalization of the Poincaré index (which also analyzes the behavior of the flow on the boundary) and has a vast range of applications to the study of dynamical systems, e.g. the proof of chaotic behavior and bifurcation theory.

Since all $CH_i(N)$ can be arbitrary finite generated Abelian groups, a complete classification of all possible Conley indices is impossible. Important ones are

Conley index	Flow is equivalent to
$CH_*(x) = (\mathbb{Z}, \{0\}, \{0\}, \{0\})$	Attracting fixed point
$CH_*(x) = (\{0\}, \mathbb{Z}, \{0\}, \{0\})$	Fixed point with one-dimensional unstable manifold
$CH_*(x) = (\{0\}, \{0\}, \mathbb{Z}, \{0\})$	Fixed point with two-dimensional unstable manifold
$CH_*(x) = (\{0\}, \{0\}, \{0\}, \mathbb{Z})$	Repelling fixed point
$CH_*(\Gamma) = (\mathbb{Z}, \mathbb{Z}, \{0\}, \{0\})$	Attracting closed streamline
$CH_*(\Gamma) = (\{0\}, \mathbb{Z}, \mathbb{Z}, \{0\})$	Saddle-like closed streamline
$CH_*(\Gamma) = (\{0\}, \mathbb{Z}_2, \mathbb{Z}_2, \{0\})$	Twisted saddle-like closed streamline
$CH_*(\Gamma) = (\{0\}, \{0\}, \mathbb{Z}, \mathbb{Z})$	Repelling closed streamline
$CH_*(\emptyset) = (\{0\}, \{0\}, \{0\}, \{0\})$	Empty set

The rank of $CH_i(N)$ is called the i-th Betti number. In particular the Poincaré index is the alternating sum of the Betti numbers:

$$index(N) = |(CH_0(N))| - |(CH_1(N))| + |(CH_2(N))| - |(CH_3(N))| \quad (8)$$

It should be noted, that a non-trivial Conley index does not guarantee the presence of the corresponding isolated invariant set, but it indicates, that flow inside the isolating neighborhood is equivalent to it. Similarly, a trivial Conley index, which means that all groups are identical $\{0\}$, could just mean, that the included invariant sets are cancelling each other out.

A nontrivial Conley index always implicates the existence of at least one isolated invariant set inside the isolating neighborhood. This statement is also known as the *Wazewski property*.

Readers who would like to know more about the computation of homology are referred to [16, 18]. Readers who are not familiar with homology may also have a look at [12]. Efficient implementations of homology algorithms already exist [2], so we do not need to concern about these issues.

Recent publications [20, 21] have particulary shown interest in visualizing closed streamlines of saddle-like behavior. Figure 6 shows that our algorithm is able to extract an artificially generated one. Peikert et al. were able to find a twisted saddle-like closed streamline from a CFD simulation of a Pelton turbine.

To the best of our knowledge, not much is explored about the extraction of compact invariant 2-manifolds from fluid flows, e.g. on an invariant torus (subject of publication [24]) almost every case from gradient-like flow to chaotic behavior may exist.

7   Results

We applied both techniques from Sects. 4 and 5 to several artificially generated data and CFD-simulation datasets, Figs. 3, 5 and 6 are just a selection.

Where the results of the geometry-based Morse decomposition become more coarse when the swirl of the field increases, the streamline-based version can compensate this problem partially by raising the maximum arclength of integration. This will lead to a higher computation time, of course. It seems like that there
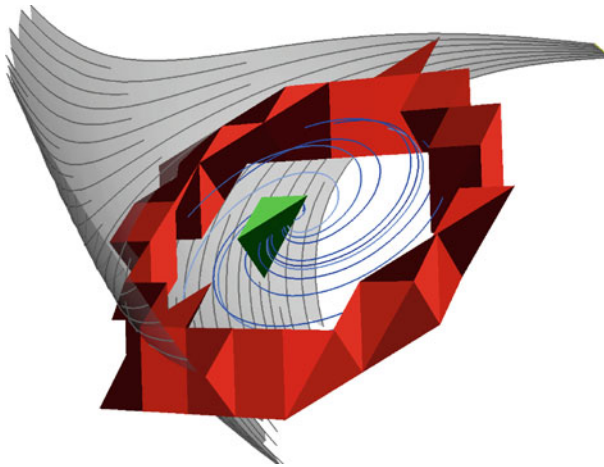
Fig. 6 A saddle-like closed streamline extracted by the streamline-based Morse decomposition. Such a feature cannot be found be arbitrarily placed individual streamlines [30]. The stream surface (*grey*) indicates a divergent behavior. But inside the stable manifold, which is a plane, where the *blue* trajectories are placed, it is also connected with a fixed point that is a source. So the extracted Morse set of red tetrahedra must act as a saddle

exists a ideal integration range for each field, because the extracted structures cannot become thinner than the diameter of a cell.

A gas furnace chamber (Fig. 5) poses a real challenge due to highly turbulent flow and is recorded by the following table. All computations were done by a single core CPU with 2.4 GHz:

Cells	Arclength of integration	Time to process	Morse sets	Number of arcs in the MCG
31,881	5	490 s	31	104
31,881	10	1,136 s	36	107
31,881	15	1,767 s	38	87

8 Conclusions and Future Work

We have shown two approaches to a Morse decomposition in three dimensions and different possibilities to classify the obtained Morse sets. We found out in our experiments, that for a large fixed integration length, there are still Morse sets remaining, that cannot be further decomposed, so there is plenty of potential in improving the streamline-based algorithm, i.e., repeatedly applying it to the remaining sets with increasing arc-length parameter. In addition, a parallelization must be considered as a must in future work. The main difference to classical topology is, that the equivalence classes of streamlines are not induced by having the same ω- and α- limit set by integration, but being in the same strongly connected component. Though the complex shape of Morse sets has theoretically a higher variation in three dimensions, practically fixed points of saddle character will dominate in data obtained from CFD simulations. The clusters of cells do not always give an immediate insight into the behaviour of the field, but it still can be used as a preprocessing algorithm for finer techniques. An interesting challenge in the future is to make general conclusions in how exactly the size of the grid and the length of integration will influence the results.

References

1. Boczko, E., Kalies, W., Mischaikow, K.: Polygonal approximation of flows. Topology Appl. **154**, 2501–2520 (2007)
2. Computer Assisted Proofs in Dynamics group, (2011) http://capd.ii.uj.edu.pl,

3. Chen, G., Mischakow, K., Laramee, R.S., Pilarczyk, P.: Vector field editing and periodic orbit extraction using morse decomposition. IEEE Trans. Visual. Comput. Graph. **13**, 769–785 (2007)
4. Chen, G., Mischakow, K., Laramee, R.S.: Efficient morse decompositions of vector fields. IEEE Trans. Visual. Comput. Graph. **14**, 848–862 (2008)
5. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing. Prentice Hall, NJ (1999)
6. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical morse complexes for piecewise linear 2-manifolds. Proceedings of the seventeenth annual symposium on Computational Geometry, pp. 70–79 (2001)
7. Forman, R.: Combinatorial vector fields and dynamical systems. Mathematische Zeitschrift **228**, 629–681 (1998)
8. Gansner, E.R., Koutsofios, E., North, S.C., Vo, K.-P.: A technique for drawing directed graphs. IEEE Trans. Software Eng. **19**(3), 214–230 (1993)
9. Guylassy, A., Bremer, P., Hamann, B., Pascucci, V.: A practical approach to morse-smale complexes for three dimensional scalar functions. IEEE Trans. Visual. Comput. Graph. **14**(6), 1619–1626 (2008)
10. Hairer, E., Syvert, P., Wanner, G.: Solving Ordinary Differential Equations I: Nonstiff Problems. Springer, Berlin (2008)
11. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional flows. Physica D **149**, 248–277 (2001)
12. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
13. Helman, J., Hesselink, L.: Visualizing vector field topology in fluid flows. IEEE Comput. Graph. Appl. **11**, 36–46 (1991)
14. Hirsch, M., Smale, S., Devaney, R.: Differential Equations, Dynamical Systems and An Introduction to Chaos. Elsevier, New York (2004)
15. Hultquist, J.: Constructing stream surfaces in steady 3-D vector fields. Proceedings IEEE Visualization 1992, pp. 171–178, Boston, MA (1992)
16. Kaczynski, T., Mischaikow, K., Mrozek, M.: Computational Homology. Springer, Berlin (2003)
17. Kalies, W., Ban, H.: A computational approach to Conley's decomposition theorem. J. Comput. Nonlinear Dyn. **1**(4), 312–319 (2006)
18. Mischaikow, K.: The Conley index theory: a brief introduction. Banach Cent. publ. **47** (1999)
19. Mrozek, M., Zgliczynski, P.: Set arithmetic and the enclosing problem in dynamics. Annales Polonici Mathematici **74**, 237–259 (2000)
20. Peikert, R., Sadlo, F.: Flow topology beyond skeletons: visualization of features in recirculating flow. Topology-Based Methods in Visualization II, Springer, Berlin, pp. 145–160 (2008)
21. Peikert, R., Sadlo, F.: Topologically relevant stream surfaces for flow visualization. Proceedings of Spring Conference on Computer Graphics, pp. 171–178 (2009)
22. Post, F., Vrolijk, B., Hauser, H., Laramee, R.S., Doleisch, H.: The state of art in flow visualization: feature Extraction and tracking. Comput. Graph. Forum **22**(4), 775–792 (2003)
23. Reininghaus, J., Hotz, I.: Combinatorial 2D vector field topology extraction and simplification. Topology in Visualization (2010)
24. A.R. Sanderson, G. Chen, X. Tricoche, D. Pugmire, S. Kruger, J. Breslau: Analysis of Recurrent Patterns in Toroidal Magnetic Fields, In Proceedings Visualization / Information Visualization 2010. IEEE Transactions on Visualization and Computer Graphics, **16**(6) (2010)
25. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J.Comput. **1**, 146–160 (1972)
26. Theisel, H., Weinkauf, T., Hege, H.-C., Seidel, H.-P.: Saddle Connectors - An Approach to Visualizing the Topological Skeleton of Complex 3D Vector Fields, Visualization Conference, IEEE, 0,30 (2003)
27. Theisel, H., Weinkauf, T., Hege, H., Seidel, H.: Grid independent detection of closed streamlines in 2D vector fields. Proceedings of Vision, Modeling, and Visualization 2004 (2004)

28. Tricoche, X., Scheuermann, G., Hagen, H.: A Topology simplification method for 2D vector fields. IEEE Visualization 2000 Proceedings, pp. 359–366 (2000)
29. Weiskopf, D., Erlebacher, B.: Overview of flow visualization. The Visualization Handbook, pp. 261–278. Elsevier, Amsterdam (2005)
30. Wischgoll, T., Scheuermann, G.: Detection and visualization of planar closed streamlines. IEEE Trans. Visual. Comput. Graph. **7**, 165–172 (2001)
31. Zhang, E., Mischaikow, K., Turk, G.: Vector field design on surfaces. ACM Trans. Graph. **25**, 1294–1326 (2006)

Part II
Hierarchical methods for extracting and visualizing topological structures

Topological Cacti: Visualizing Contour-Based Statistics

Gunther H. Weber, Peer-Timo Bremer, and Valerio Pascucci

1 Introduction

Isosurfaces [10, 13, 14] are an extremely versatile and ubiquitous component of data analysis since distinct isosurfaces often have a useful physical interpretation directly related to the application domain. In premixed combustion simulations, for example, an isotherm (isosurfaces of temperature) of the appropriate temperature is often associated with the location of the flame; in molecular simulations, isosurfaces of a certain charge density indicate molecular boundaries; and in fluid simulations, isosurfaces can be used to identify an interface between mixing fluids. Furthermore, isosurfaces provide a geometric representation of scientific data that supports the definition of further derived quantities (such as surface area) that are useful for an in-depth analysis.

Due to their importance, isosurfaces and in particular their topology have been the focus of an extensive body of research. In particular, the dependence of isosurface properties as a function of the isovalue has been examined in detail. Traditionally, there are two complementary types of information derived to aid in identifying interesting and relevant isosurfaces. Structural approaches, often based

G.H. Weber (✉)
Computational Research Division, Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720, USA
e-mail: GHWeber@lbl.gov

P.-T. Bremer
Center of Applied Scientific Computing, Lawrence Livermore National Laboratory, Box 808, L-560, Livermore, CA 94551, USA
e-mail: bremer5@llnl.gov

V. Pascucci
Scientific Computing and Imaging Institute, University of Utah, 72 South Central Campus Drive, Salt Lake City, UT 84112, USA
e-mail: pascucci@sci.utah.edu

on the contour tree, provide information about individual contours, the connected components of isosurfaces, and how their number changes as the isovalue varies. The other class of approaches, including the contour spectrum, derive quantitative measurements, such as surface area from the isosurfaces, and plot these quantities as a function of isovalue. While providing quantitative information about the isosurface, information about individual connected components is lost, and these approaches usually result in a single one-dimensional function plot.

It is possible to combine these two complementary techniques by considering the segmentation implied by the contour tree. Each arc in the contour tree corresponds to a region swept by the contours represented by the arc. Using this segmentation, it is possible to calculate measurements on a per-contour basis. In this paper, we introduce *topological cacti*, a new visualization metaphor for the contour tree that incorporates these segmentation-based quantitative measurements into the graph display. Like the toporrery [15], topological cacti utilize a radial graph layout in three dimensions to avoid self-intersections. However, topological cacti also support the display of quantitative measures along with the graph. We draw graph edges of the toporerry as cylindrical extrusions and map a selected quantity to graph edge radius. We augment this visualization by adding spikes to display a second quantity, thus completing the cactus metaphor.

We will now describe how to display topological cacti. As a first step, we calculate measurements based on the segmentation implied by the contour tree (Sect. 3). In particular, we compute the contour spectrum for individual contours to provide per-contour area and volume measures (Sect. 3.1). Subsequently, we describe how to use this information in layout computation of topological cacti (Sect. 4). We illustrate the concept of topological cacti on example data sets using the per-contour representation of the contour spectrum (Sect. 5). We also demonstrate the display of other derived quantities, such as those associated with a join/merge tree resulting from the analysis of combustion simulations, as well as merge trees derived from higher-dimensional data.

2   Background and Related Work

2.1   Contour Trees

The contour tree [2] is a special case of the more generally defined Reeb graph [16] and captures the topological evolution of an isosurface as the isovalue varies. Critical points, where the number of contours changes, appear as nodes in the contour tree. Nodes of degree one (leaves of the tree) correspond to extrema, where contours are created or destroyed. Interior nodes of degree three or higher correspond to "saddles," where two or more contours merge or split. Arcs of the contour tree represent contours between critical points, i.e., contours that do not change topology (with the exception of genus changes) as the isovalue varies between critical values.

Conceptually, one can compute the contour tree by considering an isosurface for a particular isovalue and collapsing each contour (connected component) into a single point. Performing this process for each isovalue yields the graph structure described in the previous paragraph. However, currently most algorithms for computing the contour tree are based on the work by Carr et al. [4] and take a different, computationally more efficient approach. Instead of computing the contour tree directly, this algorithm starts by computing two simpler graphs, the join (or merge) tree and the split tree. The join tree tracks how connected regions above a threshold join as the threshold is decreased, and it can be computed efficiently using a union-find approach. The split tree analogously traces the behavior of regions below a threshold as that threshold is increased. A second phase of the algorithm uses join and split tree to compute the contour tree using graph-based operations. While join trees and split trees have been introduced in the context of contour tree calculation, they are also useful as structures in their own right. For example, they represent all possible segmentations based on thresholding and enables extensive parameter studies based on those segmentations [3].

Contour trees of even moderately-sized data sets can be quite complex. This complexity can be due to either noise or the presence of features at various scales. To remove noise, or features at smaller scale, it is necessary to simplify the contour tree. Carr et al. [5] simplified the contour tree using two operations: an arc-pruning operation to remove an extremum by "chopping" off a branch of the contour tree, and a vertex collapse to remove the resulting degree two node. Local geometric measures, like persistence (difference in function value) or volume, define an order in which arcs are pruned, i.e., what isosurface features are deemed less important and are removed first. Takahashi et al. [19,20] use the related volume skeleton tree to simplify topology in a similar fashion. In subsequent work, they utilized the results, e.g., for the design of transfer functions that emphasize topology and take contour nesting properties into account [21].

Pascucci et al. [15] proposed the branch decomposition of the contour tree as an efficient data structure for encoding a multi-resolution representation of the contour tree. This representation decomposes the contour tree into a set of branches stored as a rooted tree. A root branch connects a minimum-maximum pair in the contour tree. Other extrema are ordered by an importance metric (e.g., persistence), and child branches connect them to a saddle in an existing branch. In the resulting tree, each extremum appears as part of a branch, and the level in which a branch appear corresponds to its importance. The less important a branch is, the further down in the tree it appears. Using this data structure, it is possible to traverse the contour tree efficiently up to a desired level of detail (as specified by the importance metric).

2.2 Contour Tree Visualization

The contour tree is a graph and it is possible to display it using standard graph drawing algorithms like those provided by the GraphViz package [8]. Traditional layouts of the contour tree usually constrain the height of nodes to correspond to

function value. This layout has the advantage that a horizontal line drawn at the height for a given isovalue intersects as many edges (or nodes) in the contour tree as there are contours for that value. This property significantly increases readability of the graph. More recently, Heine et al. [9] presented a fast and effective technique for contour tree layout in the plane.

One disadvantage of the height constraint is that it is not always possible to layout the contour tree without self-intersection. To avoid this problem, Pascucci et al. [15] proposed an alternative three-dimensional radial layout for the contour tree. This layout is motivated by using an orrery, i.e., a mechanical device that illustrates relative positions of planets and moons in the solar system, as a metaphor. Extrema take the place of planets and moons, and the toporrery shows the hierarchical relationship between the branches connecting them.

The toporrery preserves the contour tree as a graph, and its layout is related to traditional radial graphic layouts. As such, it does not provide information beyond the graph structure of the contour tree. An alternate approach uses structural information to illustrate the nesting behavior of contours [18]. Mizuta et al. [11, 12] computed contour trees on digital image data and visualized the resulting trees using the contour nest. The contour nest focuses on the nesting properties of isosurfaces and represents the contour tree as a set of nested rectangles. In this visualization, rectangle size corresponds to feature size (area in 2D and volume in 3D). However, the contour nest displays only a global value for each branch and does not show how measures vary as the isovalue of a contour changes.

Topological landscapes [22] utilize another metaphor to visualize a contour tree. Observing that the contour tree was originally introduced in the context of two-dimensional terrains and that a terrain can have the topology implied by a contour tree, this metaphor constructs a terrain that has the same topology as the contour tree. In this way, the terrain serves as a proxy representation for a higher-dimensional (three dimensions or more) data set. To improve the expressiveness of the terrain metaphor, it is possible to re-parametrize the terrain in such a way that the area of features corresponds to a specified metric (e.g., volume associated with a branch of the branch decomposition). Like the contour nest, the construction scheme is limited to visualizing on the branch level, i.e., variations within a branch are ignored.

2.3 Contour Spectrum

While the contour tree provides structural information about an isosurface and the behavior of individual components, it does not provide quantitative information. The contour spectrum [1] on the other hand provides global measures for an isosurface as a function of isovalue. Observing that the surface area of an isosurface in a tetrahedral mesh can be expressed as a B-Spline, the authors develop a framework that encodes total surface area, exterior and interior volume, as well as gradient magnitude.

Fundamentally, the authors use the connection between isosurfaces on a tetrahedral mesh and simplex splines. Based on this relationship, they derive a B-Spline

representation of isosurface area as a function of isovalue within an individual tetrahedron of the data set. Combining the B-Spline functions of all individual tetrahedra of a given mesh results in a B-Spline representation of the entire data set. The authors subsequently derive a second B-Spline describing the volume enclosed in the isosurface by integrating the area B-Spline.

3 Computing per-Component Measures

To display per-component information about an isosurface, we need to compute it and associate it with the contour tree. For the purpose of computing these measures, we extend the contour spectrum to provide per-component measures (Sect. 3.1) or calculate measures based on the segmentation implied by the contour tree (Sect. 3.3). While prior work uses, e.g., interval volumes [7], to derive measures such as area and volume for contours (i.e., connected isosurface components), our approach provides these measures as a function of isovalue that is associated with contour tree edges. This function representation makes it possible to compute an individual, contour-specific area or volume for any isovalue along an edge, as opposed to approximating a single volume for an entire edge like prior work on topology simplification [19, 21].

In this context, it is beneficial to use the branch decomposition of the contour tree. (1) Branches in this decomposition may combine several arcs of the contour tree and therefore the segmentation consists of fewer regions. Nevertheless, each contour maps uniquely to a unique branch and a value along the branch. (2) The hierarchical nature of the branch decomposition simplifies finding paths in the contour tree. (3) The topological cactus layout is based on the toporrery layout, which is inherently based on the branch decomposition.

3.1 Computing the Contour Spectrum

We compute per-component measures of area and volume using the contour spectrum. Computing the area is relatively straightforward, as there are no ambiguities. Associating volume with the contour tree is more difficult, since for nested contours the definition of enclosed volume is ambiguous. For both measurements, we associate either a B-Spline or a piecewise polynomial function with each branch of the branch decomposition.

3.2 Computing per-Contour Area

Calculating the contour spectrum globally relies on the fact, that within a tetrahedral cell surface area can be expressed as a function of isovalue as a B-Spline. In the

original contour spectrum algorithm, contributions of all tetrahedra are accumulated into a global B-Spline. To compute area on a per-component basis, we associate a B-Spline (or alternatively a piecewise polynomial function) with each branch of the branch decomposition, and accumulate contributions per-component.

For accumulating per-component area, we observe that linear tetrahedra do not contain any critical points in their interior. In particular, they do not contain any saddles. Consequently, the "contour tree" for the individual tetrahedra is a straight line, ranging from its minimum vertex to its maximum vertex. Thus, in the contour tree a tetrahedron corresponds to a single path between two vertices (not necessarily nodes), along which the function value uniquely identifies the contour containing this tetrahedron. As a result, we can consider each tetrahedron and accumulate its contributions to the branch decomposition.

We perform this accumulation as follows. First, we compute contour tree and branch decomposition. We also compute the segmentation of the data set implied by the contour tree. This segmentation is stored as an array with an entry for each sample of the data set that contains a reference to the single branch within the decomposition that corresponds to the sample.

Subsequently, we accumulate the contributions of all tetrahedra to the contour spectrum. For each tetrahedron, we first determine its minimum and its maximum vertex. These vertices define the path in the contour tree along which contributions have to be accumulated, as shown in Fig. 1. We use the segmentation information to identify the branches corresponding to the minimum and maximum value.

We then find the path in the branch decomposition that connects the minimum and the maximum. Since the branch decomposition is a rooted tree (of branches), we simply find the common ancestor branch of the minimum and maximum vertex to construct their connecting path, see Fig. 1. In the figure, the root branch is the common branch, the path from the maximum to the root traverses two branches and the path from the minimum traverses one branch. Therefore, two lists of branches descending from the minimum and the maximum, as well as the common branch
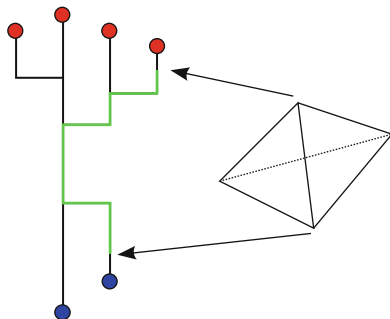
Fig. 1 Computing the contour spectrum utilizing the branch decomposition. A linear tetrahedron corresponds to a path in the branch decomposition. After identifying this path connecting the tetrahedron's maximum and minimum vertex, we add appropriate contributions to the contour spectrum associated with each branch

connecting the last branches in these lists, completely define the path in the branch decomposition.

Subsequently, we split the B-Spline or piecewise polynomial function into segments corresponding to the branches along the path. For the first branch from the maximum, this segment corresponds to the range from the maximum tetrahedron value to its saddle. For all subsequent branches along the path, the segment corresponds to the range from the saddle value of the previous branch along the path to the saddle value of the current branch. Segments for the minimum path are defined analogously. Finally, the segment along the branch connecting the two paths corresponds to the range between the two saddle values.

3.2.1 Computing per-Contour Volume

Computing per-contour volume is more difficult. If a contour contains another contour, it is ambiguous, whether the volume of the nested contour should be subtracted or not. To avoid ambiguities, we follow the convention that we calculate volume from the last saddle. Thus, the volume associated with a contour corresponds to the volume swept by that contour as long as it "maintained its identity," i.e., since its last "interaction" with another contour. Adding the contribution to the branch decomposition is similar to adding the contribution for the area. There are two key differences: Instead of stopping at the maximum value of the tetrahedron, the contribution on the maximum branch reaches until the next saddle value. This difference is due to the fact that once the maximum value of the tetrahedra is passed the volume function is a constant corresponding to the volume of the tetrahedron. Furthermore, following our convention, for each segment, we subtract the volume at the saddle from the entire function, such that for each segment only the volume swept since the saddle is considered.

Unlike the original contour spectrum approach, we also observe that obtaining the volume by integrating the area yields incorrect results. Expressing the volume as integral of area corresponds to a substitution of the integration variable, and we need to multiply the result by the derivative, which corresponds to multiplication with the inverse gradient magnitude. This is analogous to observations on computing other isosurface statistics [17].

3.3 Segmentation-Based Measures

Visualizing volume and area already enhances contour tree display and provides valuable information about a function. However, in many application areas, it is desirable to display additional statistics, such as the mean value of a second variable on the isosurface. We compute these values at discrete locations of the contour tree or the join tree. To compute these values, we consider the segmentation implied by the contour tree [23] or the join tree [3] and accumulate these statistics for all

samples corresponding to a branch or arc. This calculation would yield a single value per branch. To approximate function behavior along a branch or arc, we split it at regular intervals and insert degree two nodes into the join tree. When computing the branch decomposition, we associate these function values with the branch. This approach basically approximate calculating statistics with a segmentation defined in interval volumes [19].
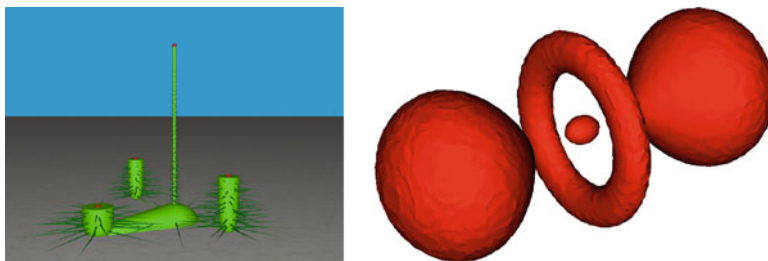
4   Topological Cacti Layout

For generating topological cacti, we use an radial graph layout identical to that of the toporrery [15]. Extrema are ordered in concentric discs, where the disc is chosen based on the depth of an extremum in the branch decomposition. Like in the original toporrery, the radius ratio is constant in our layout, and each branch is assigned a radial wedge with an angle proportional to the number of child branches. To display metrics, we draw each branch as cylindrical extrusion and chose the radii according to a first selected measure (clamping however at a minimum radius to ensure that branches do not disappear). We draw connectors as sphere sweeps to ensure smooth connection of the cylindrical extrusion corresponding to parent and child branch. Before mapping a metric such as volume to radius, we support choosing a variety of transformation functions (square root, cube root, logarithm) to ensure that higher-dimensional properties are appropriately scaled to the one dimensional radius. This is analogous to a similar transformation before mapping volume to area in the topological landscapes [22].

For display of a second metric, we augment the cylindrical extrusions with spikes. We start by placing circles along regular intervals of the cylindrical extrusion and choosing angles in regular intervals along this circle. We then choose spike location by jittering these regular samples, i.e., we add random offset angles and heights to the originally chosen spike locations. We then calculate the metric value for the function value corresponding to the (jittered) height. Spikes are drawn as cones with a fixed (user defined) base radius and a length depending on the metrics. Similar to the extrusion radius, we support user chosen mappings such as square root, cube root or logarithm.

5   Results

Figure 2 shows an example of a topological cactus for the hydrogen atom data set (simplified to a persistence of ten). Branch width corresponds to the cube root of volume and spike length to the square root surface area. The root branch corresponds to the evolution of the contour in the center. It is immediately obvious that despite its very high persistence (resulting in it being chosen as root branch), both its volume and surface area are very small. The two lobes, corresponding to the two

Fig. 2 Topological cactus for the hydrogen atom (*left*) and isosurface rendering of the corresponding data set (*right*). The root branch in the center corresponds to the contour in the center of the ring. The two symmetric branches correspond to the two lobes, and the lowest branch to the ring. The branch width corresponds to the cube root of enclosed volume, and spike length to the square root of surface area
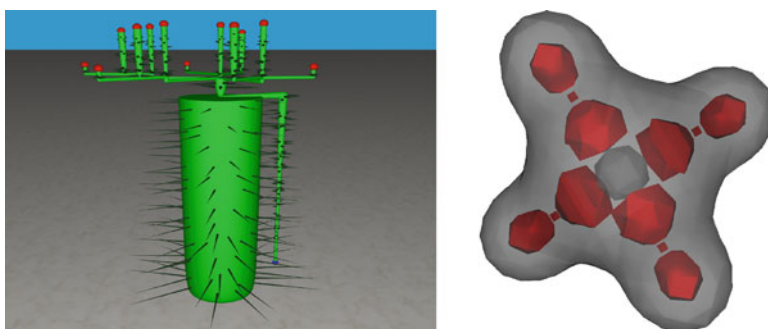
Fig. 3 Topological cactus for the methane molecule (*left*) and isosurface rendering of the corresponding data set (*right*). The right hand image shows isosurfaces for two isovalues. The *red* isosurface corresponds to isovalue close to the maximum value and shows contours created around the 12 maxima. The *gray* isosurface corresponds to a lower isovalue at a level where all 12 red contours have merged and then split into two separate components. In the cactus display, branch width corresponds to the cube root of volume and spike length to the square root of area

symmetric branches are much larger in both volume and surface area. Finally, the ring, corresponding to the lowest branch, has the largest volume and area.

Figure 3 shows the topological cactus for the methane data set. The image on the right hand side shows corresponding isosurfaces. The red isosurface corresponds to a level close to the maxima and consists of 12 contours that are created around the 12 maxima visible on the cactus. As the cactus view shows, the three contours of each "arm" merge first, and subsequently these four "arms" merge into a single contour. At a lower isovalue, this contour splits into two separate contours. The cactus shows, how the volume of these components varies with the isovalue. For example, surface area and volume of the outer component remain relatively constant, while the inner component shrinks slowly and disappears.

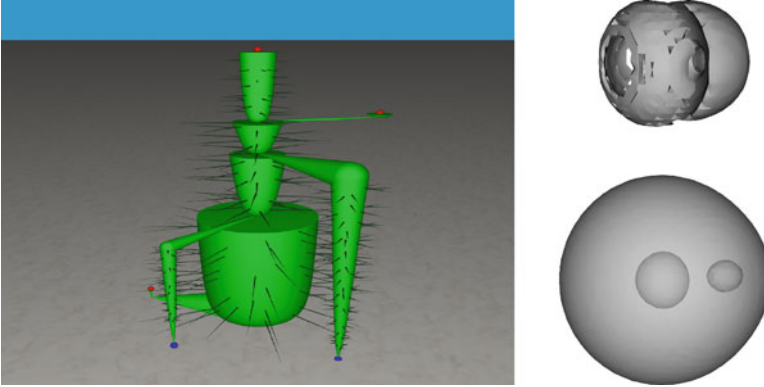Figure 4 shows the nucleon data set (simplified to a persistence of three). The right hand side shows isosurfaces at two levels (in separate images due to

Fig. 4 Topological cactus for the nucleon (*left*) and isosurface rendering of the corresponding data set (*right*). Branch width corresponds to the cube root of volume, and spike length to the square root of surface area. The cactus only shows branches with a minimum persistence of three (data values are quantized to byte values)

complexity). The lower isosurface image shows contours just below the second lowest saddle resulting in three separate contour. We note that for any value, the contour branching off at the lowest saddle is too small to be visible in an rendered isosurface image. The cactus representation also clearly shows that volume and area for this contour are minute. The upper isosurface shows contours for an isovalue slightly above the higher saddle. The cactus shows clearly that even though one of the components has only small persistence, it has actual a volume comparable to the other component.

Figure 5 shows the topological cactus of the merge tree of fuel consumption rate in a combustion simulation. Branch width corresponds to the cube root of number of vertices, thus approximating volume. Spike length corresponds to the variance of temperature. Each branch corresponds to a burning region around a maximum of fuel consumption. The cactus shows that there are two types of regions: (1) Regions that are relatively wide close to the maximum and thus are comprised of a relatively large number of vertices and (2) smaller regions with a thin region around the maximum. The latter occur at lower fuel-consumption levels. The spikes show that temperature variance is small for high fuel consumption, corresponding to intense burning and is (expectedly) larger for fuel consumption corresponding to non-burning regions.

Figure 6 shows the cactus of the merge tree of a 21 dimensional climate simulation ensemble and the more traditional graph layout of the branch decomposition of the merge tree. The data set consists of 1,197 climate simulations performed with the Community Atmospheric Model (CAM) [6] each using a different combination of 21 input parameters. Such ensembles are used to study the sensitivity of the climate predictions to changes in input parameters. The output function shown in Fig. 6 is the average longwave energy output (FLUT) in the winter months (December,
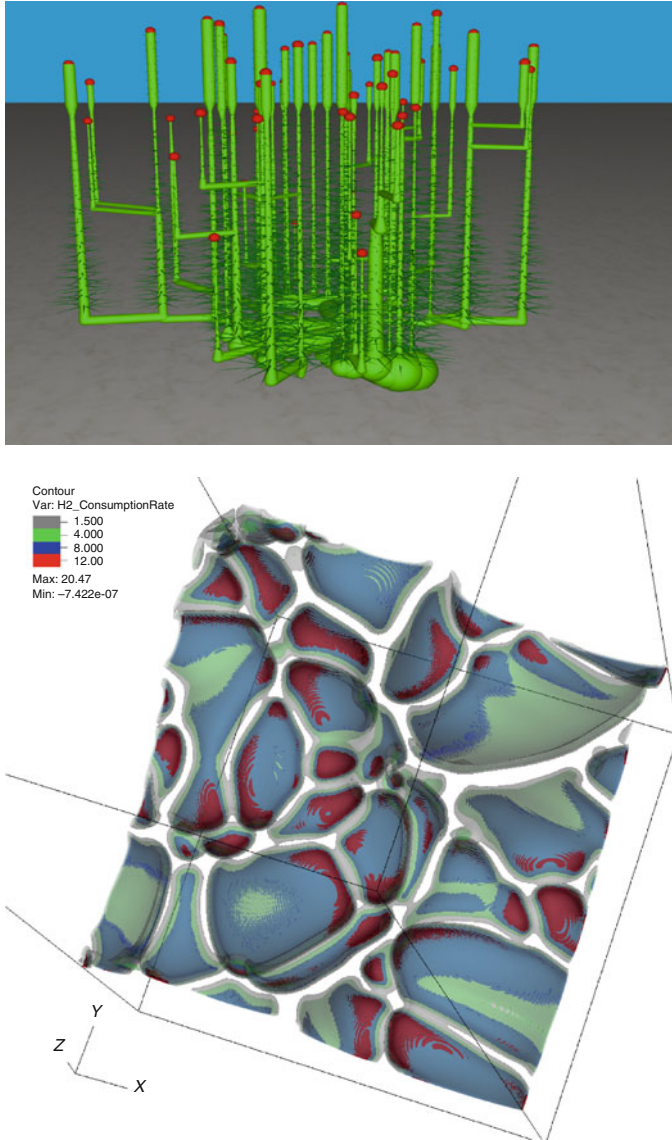
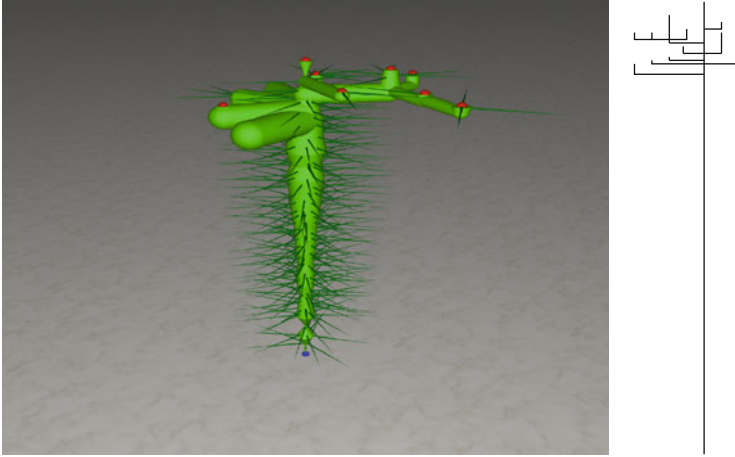**Fig. 5** Topological cactus of the merge tree of fuel consumption rate in a combustion simulation (*top*). The width corresponds to the cube root of number of vertices in a region. Spike length shows the variance of temperature. Isosurfaces for four different fuel consumption rates illustrating burning regions for these thresholds (*bottom*)

Fig. 6 Topological cactus of a climate simulation showing the merge tree of long wave energy output in the winter months (*left*) and graph layout of the branch decomposition (*right*). The width corresponds to cube root of vertex count. The spike length shows the mean value of FLUT

January, February). Unlike the traditional graph layout, the cactus shows that most of the "volume" of the parameter space is collected in several large but relatively low persistent branches of maxima. This fact is interesting as the observed energy output is expected to be closer to the global minimum. Thus, it appears that only a small portion of the input samples produces realistic outputs. Nevertheless, all persistences are fairy low indicating that the overall behavior is stable under perturbation of the input parameters.

6 Conclusions and Future Work

We have introduced topological cacti and demonstrated how they combine insights obtained from the contour tree with metrics derived on a per contour bases. In particular, where importance measured by persistence differs from other metrics such as area of volume, this combination can greatly enhance the usefulness of contour tree representations. Going forward, we plan to improve both contour spectrum and cactus representation. Our current approach of computing the contour spectrum in piecewise polynomial/B-Spline representation works well for quantized data sets. However, for real-valued data, each data value appears in the knot vector, making this representation inefficient. To counteract this problem, we need to devise a simplification scheme that can be applied during the calculation and still support error bounds.

For the cactus representation, we plan to improve the way connectors are drawn. Drawing them as sphere sweeps sometimes hides details about the width of the

parent branch. Furthermore, we plan to improve the placement of spikes to make their distribution more even, e.g., by increasing the number of spikes per angle for very wide branches (i.e., where the first metric is large).

References

1. Bajaj, C., Pascucci, V., Schikore, D.R.: The contour spectrum. In: Proc. IEEE Visualization, published in October by IEEE Computer Society Press, pp. 167–175 (1997)
2. Boyell, R.L., Ruston, H.: Hybrid techniques for real-time radar simulation. In: Proc. 1963 Fall Joint Computer Conference, pp. 445–458. IEEE, NY (1963)
3. Bremer, P.T., Weber, G.H., Tierny, J., Pascucci, V., Day, M.S., Bell, J.B.: A topological framework for the interactive exploration of large scale turbulent combustion. In: Proc. 5th IEEE International Conference on e-Science, pp. 247–254. Oxford, UK (2009)
4. Carr, H., Snoeyink, J., Axen, U.: Computing contour trees in all dimensions. Comput. Geom. Theor. Appl. **24**(2), 75–94 (2003)
5. Carr, H., Snoeyink, J., van de Panne, M.: Simplifying flexible isosurfaces using local geometric measures. In: Proc. IEEE Visualization 2004, published in October by IEEE Computer Society Press, pp. 497–504 (2004)
6. Collins, W.D., Rasch, P.J., Boville, B.A., Hack, J.J, McCaa, J.R., Williamson, D.L., Briegleb, B.P, Bitz, C.M., Lin, S.-J., Zhang, M.: The formulation and atmospheric simulation of the community atmosphere model version 3 (CAM3). J. Climate **19**, 2144–2161 (2006). doi: 10.1175/JCLI3760.1
7. Fujishiro, I., Maeda, Y., Sato, H., Takeshima, Y.: Volumetric data exploration using interval volume. IEEE Trans. Vis. Comp. Graph. **2**(2), 144–155 (1996)
8. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. Software Pract. Ex. **30**, 1203–1233 (1999)
9. Heine, C., Schneider, D., Carr, H., Scheuermann, G.: Drawing Contour Trees in the Plane. IEEE Trans. Vis. Comp. Graph. **17**(11), 1599–1611 (2011)
10. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm. Comput. Graph. (Proc. ACM SIGGRAPH 87) **21**(4), 163–169 (1987)
11. Mizuta, S., Suwa, Y., Ono, T., Matsuda, T.: Description of the topological structure of digital images by contour tree and its application. Tech. rep., Institute of Electronics, Information and Communication Engineers (2004)
12. Mizuta, S., Ono, T., Matsuda, T.: Contour nest: A two-dimensional representation for three-dimensional isosurfaces. In: Proc. Volume Graphics, Published July 2006 by the Eurographics Association, Aire-la-Ville, Switzerland, pp. 67–70 (2006)
13. Montani, C., Scateni, R., Scopigno, R.: A modified look-up table for implicit disambiguation of marching cubes. Vis. Comput. **10**(6), 353–355 (1994)
14. Nielson, G.M.: On marching cubes. IEEE Trans. Vis. Comp. Graph. **9**(3), 341–351 (2003)
15. Pascucci, V., Cole-McLaughlin, K., Scorzelli, G.: The Toporrery: Computation and Presentation of Multi-Resolution Topology, pp. 19–40. Springer, Berlin (2009)
16. Reeb, G.: Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique. Comptes Rendus de l'Acadèmie des Sciences de Paris **222**, 847–849 (1946)

17. Scheidegger, C.E., Schreiner, J.M., Duffy, B., Carr, H., Silva, C.T.: Revisiting histograms and isosurface statistics. IEEE Trans. Vis. Comp. Graph. **14**, 1659–1666 (2008)
18. Shinagawa, Y., Kunii, T.L., Kergosien, Y.L.: Surface coding based on morse theory. IEEE Comput. Graph. Appl. **11**(5), 66–78 (1991)
19. Takahashi, S., Takeshima, Y., Fujishiro, I.: Topological volume skeletonization and its application to transfer function design. Graph. Model. **66**(1), 24–49 (2004)
20. Takahashi, S., Nielson, G.M., Takeshima, Y., Fujishiro, I.: Topological volume skeletonization using adaptive tetrahedralization. In: Proc. Geometric Modeling and Processing, Published September 2007 by IEEE Computer Society Press, pp. 227–236 (2004)
21. Takahashi, S., Takeshima, Y., Fujishiro, I., Nielson, G.M.: Emphasizing isosurface embeddings in direct volume rendering. In: Scientific Visualization: The Visual Extraction of Knowledge from Data, pp. 185–206. Springer, Berlin (2005)
22. Weber, G.H., Bremer, P.T., Pascucci, V.: Topological landscapes: A terrain metaphor for scientific data. IEEE Trans. Vis. Comp. Graph. (Proc. Visualization 2007) **13**(6), 1416–1423 (2007)
23. Weber, G.H., Dillard, S.E., Carr, H., Pascucci, V., Hamann, B.: Topology-controlled volume rendering. IEEE Trans. Vis. Comp. Graph. **13**(2), 330–341 (2007)

Enhanced Topology-Sensitive Clustering by Reeb Graph Shattering

W. Harvey, O. Rübel, V. Pascucci, P.-T. Bremer, and Y. Wang

1 Introduction

Scalar-valued functions are abundant in scientific data. Understanding and visualizing the structure of these functions is a fundamentally important aspect of scientific research. When the domain of a scalar-valued function is low-dimensional, these two tasks are often straightforward. However, as the topological and geometric complexity of the domain increases, exploring and understanding the function can become challenging. Recently, feature-based methods using ideas from geometry and topology have proven useful for extracting meaning from high-dimensional data.

Computational efficiency is very important for data exploration, yet it can be an elusive goal when the data is high-dimensional. For example, creating a precise reconstruction of a high-dimensional domain from point cloud data and detecting high-dimensional cycles can be computationally prohibitive. Thus, a reasonable compromise is to strive for a balance between fidelity and processing speed. The fundamental idea of topology-sensitive clustering is to partition the domain into regions that are in some sense simple, capturing the essence of the topological structure of the scalar function at hand while maintaining economy of computing effort.

W. Harvey (✉) · Y. Wang
Ohio State University, 487 Dreese Lab, 2015 Neil Ave., Columbus, OH 43210, USA
e-mail: harveywi@cse.ohio-state.edu; yusu@cse.ohio-state.edu

O. Rübel · P.-T. Bremer
Lawrence Livermore National Laboratory, 422, P.O. Box 808, Livermore, CA 94551-0808, USA
e-mail: ruebel1@llnl.gov; bremer5@llnl.gov

V. Pascucci · P.-T. Bremer
Scientific Computing and Imaging Institute, School of Computing, University of Utah, 72 South
Central Campus Drive, Salt Lake City, UT 84112, USA
e-mail: pascucci@sci.utah.edu; ptbremer@sci.utah.edu

One may argue that a good clustering algorithm will partition the domain into monotone regions which are approximately topological balls, as this would provide simplicity of both the function behavior and topological structure within each region. Additionally, the accuracy of approximation should be exchanged for a significant gain in computational efficiency.

In this work, we take an additional step toward this ideal topology-sensitive clustering algorithm by extending the method of Gerber et al. [16]. Specifically, Gerber et al. [16] describes a simple discrete gradient based method to cluster points into so-called *Morse crystals* approximately analogous to the Morse-Smale complex of the input function. These crystals have simple topology (ideally a topological ball) within each of them, and have been demonstrated to be useful in understanding high dimensional scalar fields in various ways. However, in practice, these Morse crystals may have non-trivial topology inside. In this work, we take advantage of the ability to detect a certain type of non-trivial topology within each crystal, namely one-dimensional cycles (*loops*) reflected in the structure of its Reeb graph. By this property, the Reeb graph is used to subdivide a crystal into pieces which are topologically simpler. We also show preliminary experimental results to demonstrate the effectiveness of the proposed methods in several datasets.

2   Related Work

Topological methods such as the Reeb graph [29] and Morse-Smale complex [25], provide abstract representations of the fundamental structure of scalar functions. Topological structures make complex functions accessible to computational analysis and provide efficient means for defining a wide variety of features. Scalar-field topology as a general tool for analysis of scalar functions has been used in a wide array of applications ranging from, for example, physics [5, 23], biosciences [20], and medicine [7] to material sciences [18].

The Morse-Smale complex is defined as the intersection of the stable and un-stable manifolds of f, i.e., the Morse-complex of f and $-f$. The Morse-Smale complex partitions the domain into regions of monotone gradient flows, each with a single source, defined by a maximum of f, and a sink, defined by a minimum of f. In the area of visualization, computation of Morse-Smale complexes has focused mainly on n-dimensional manifolds [14,15,17]. Algorithms for computing the Morse complex – although under different names – have been proposed in a wide range of research areas. In computational geometry, the Morse complex is often described in terms of a filtration of sub-level sets of f [8,9,35]. The watershed segmentation method [4, 27], widely used in image processing, is a variant of the Morse-complex and has been described for image data [3] as well as abstract-graphs and n-dimensional grids [34]. Gradient ascent-based clustering methods, such as mean-shift [10, 12], medoid-shift [30], and quick

shift [32] clustering, are widely used in machine learning and pattern recognition, and they are also closely related to algorithms for computing and approximating the Morse complex. Our work is based on and extends a recently proposed algorithm by Gerber et al. [16] for approximating the Morse-Smale complex of unstructured, point-cloud data in n-dimensions, described in more detail in Sect. 3.2.

While the Morse-Smale complex describes the topology of a scalar function f based on the induced gradient flow, the Reeb graph [29] encodes the topology of the set of level-sets of f. The Reeb graph is constructed by contracting the connected components of the level-sets of f to points. In this work we use the Reeb graph to detect and correct loops in crystals of the approximate Morse-Smale complex. In literature, various efficient algorithm for computing the Reeb graph and its loop-free variant, the contour-tree, have been described for n-dimensional manifolds [6, 21, 28]. Here, we use the fast randomized algorithm proposed by Harvey et al. [21] to compute the augmented Reeb graph of a point set.

In order to compute the Morse-Smale complex, information about the connectivity of the domain is needed. Neighborhood/proximity graphs [22] such as the k-nearest neighbor graph and Delaunay triangulation [2] are commonly used to define neighborhoods for point cloud data. We use the approximate, undirected k-nearest neighbor graph for this purpose. We then construct the 2-skeleton of the domain, required for the Reeb graph computation, by calculating the Rips complex [33] (see Sect. 3.3.1).

3 Method

Overview. We consider the problem of understanding the structure of a scalar function defined on a high-dimensional domain $f : \mathbf{M} \to \mathbf{R}$, where $\mathbf{M} \subseteq \mathbf{R}^D$. In practice, information about the scalar function is often available only as point cloud data; i.e., a set $X = \{x_1, x_2, ..., x_n\}$ of n discrete point samples taken from the domain, and their associated set of function values $Y = \{f(x_1), f(x_2), ..., f(x_n)\}$. Thus, the objective is to acquire some insight into the structure of f based on properties which can be obtained through the analysis of X and Y. Our algorithm proceeds as follows:

1. **Domain Approximation.** The connectivity of the underlying domain, if not available, must be approximated from X. We use the approximate, undirected k-nearest neighbor graph for this purpose.
2. **Morse Crystal Decomposition.** X is partitioned into Morse crystals using the method proposed in [16].
3. **Reeb Graph Shattering.** Crystals containing loops are shattered according to the structure of their Reeb graphs.

3.1 Domain Approximation

Before any topological processing can occur, the structure of the domain must be approximated from X. The proposed method is agnostic with respect to how the approximation is performed; any suitable method which attempts to capture the neighborhood relationships of the point cloud data can be used. For the purposes of applying the technique to analysis of high-dimensional datasets while maintaining computational efficiency, we prefer to use the approximate, undirected k-nearest neighbor graph of X to represent the topology of the underlying domain. We compute this graph efficiently using the freely-available ANN library [26].

3.2 Morse Crystal Decomposition

In this section we review the Morse crystal decomposition of f, introduced briefly in [16]. We provide a detailed description of this decomposition and later show in Sect. 3.3.2 how it is augmented to yield the shattered Morse crystals.

We begin by considering the k-nearest neighbors of a point, and we describe how the function values of those neighbors give rise to the Morse crystal decomposition. Without loss of generality, henceforth we assume that all points in X have unique function values. Let x be a point of X, and let $G_{knn} = (X, E_{knn})$ be the undirected k-nearest neighbor graph of X, where $E_{knn} \subset X^2$. Let $n(x) = \{y : (x, y) \in E_{knn}\})$ denote the k-nearest neighbors of x. For any neighbor y of x,

$$m(x, y) = \frac{f(y) - f(x)}{\|y - x\|}$$

describes the approximate gradient of f at x in the direction $(y - x)$. Let $n^+(x) = \{y \in n(x) : f(x) < f(y)\}$ denote the *upper neighbors of x* and $n^-(x) = \{y \in n(x) : f(x) > f(y)\}$ denote the *lower neighbors of x*. Then the *steepest ascending neighbor x^+* and *steepest descending neighbor x^-* of x are defined as

$$x^+ = \underset{y \in (n^+(x) \cup \{x\})}{\arg\max} \; m(x, y); \qquad x^- = \underset{y \in (n^-(x) \cup \{x\})}{\arg\min} \; m(x, y).$$

The notions of steepest ascending and descending neighbors of a point lead to succinct definitions of the *steepest ascending path of x* as $\pi^+(x) = \{x\} \cup \pi^+(x^+)$ and the *steepest descending path of x* as $\pi^-(x) = \{x\} \cup \pi^-(x^-)$. Finally, let $\pi(x) = \pi^+(x) \cup \pi^-(x)$ denote the points along the discrete integral curve passing through x.

Since the Morse crystal decomposition relies on the extremal elements of $\pi(x)$ for each point $x \in X$, we will use them to define an equivalence class that gives rise to the decomposition. Let $\omega^+(x)$ and $\omega^-(x)$ denote the maximal and minimal elements of $\pi(x)$, respectively. Two points x and y are *equivalent*, denoted

$x \sim y$, if their steepest ascending paths converge to a common maximum and their steepest descending paths converge to a common minimum; that is $\omega^+(x) = \omega^+(y)$ and $\omega^-(x) = \omega^-(y)$. Then the quotient space $X_\sim$ forms the *Morse crystal decomposition of f*.

Based on these observations, it is apparent that computing $X_\sim$ admits an efficient and straightforward solution. Let F^+ and F^- be two disjoint-set forests [31], where F^+ supports $union^+(x, y)$ and $find^+(x)$ operations, and F^- supports $union^-(x, y)$ and $find^-(x)$ operations. For each point x, take $union^+(x, x^+)$ and $union^-(x, x^-)$. Then the Morse crystal membership of x is uniquely identified as the ordered pair $(find^-(x), find^+(x))$. Figure 1 illustrates the Morse crystal decomposition of point cloud data sampled from a simple two-dimensional domain.

3.2.1 Topological Simplification

To help reduce the number of spurious Morse crystals caused by sparsity in the input data or the presence of noise, the Morse crystal decomposition can be simplified by neutralizing crystals whose topological persistence is below a user-specified threshold τ. Specifically, we can use the union-find data structure to also compute the standard persistence for each minimum and maximum [13]. The output is a set $S = \{(x_i, k_i, s_i, \delta_i)\} \subset X^3 \times \mathbf{R}^+$ which encodes the persistence and pedigree of the topological components of f. Here, x_i is an extremum of a topological component with persistence δ_i, k_i is the extremum of the component which kills x_i, and s_i is the saddle point merging the two components. By using the map

$$\kappa_\tau(x_i) = \begin{cases} k_i & \text{if } \delta_i < \tau \\ x_i & \text{otherwise} \end{cases}$$

topological simplification of the Morse crystal decomposition occurs by simply using $(\kappa_\tau(find^-(x)), \kappa_\tau(find^+(x)))$ to assign points to Morse crystals.
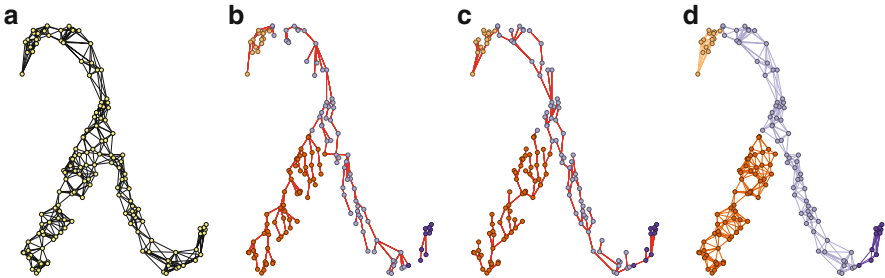
Fig. 1 An example of the Morse crystal decomposition of a simple scalar function. Here, the function value at each point is its height. (**a**) k-nearest neighbor graph of points sampled from a simply connected, lambda-shaped domain. (**b**) Edges connecting points to their steepest ascending neighbors. (**c**) Edges connecting points to their steepest descending neighbors. (**d**) The Morse crystal decomposition partitions the domain into four approximately-monotone regions

3.3 Reeb Graph Shattering

While the Morse crystal decomposition provides a useful segmentation of the
domain into approximately piecewise-monotone regions, its computational effi-
ciency trades off fidelity to the true underlying Morse-Smale complex. As a result,
a crystal may contain nontrivial loops or higher-order voids. These voids contribute
to the geometric disparity of the points within the crystal, which is antithetical to
our desired domain segmentation.

We propose to focus on the problem of finding and eliminating certain 1-loops
in Morse crystals through the process of *Reeb graph shattering*. That is, we can
further refine each Morse crystal by tracking the sequence of Reeb graph edges that
these paths visit as they traverse the crystal, and using these sequences to potentially
split the crystal into *shattered Morse crystals*. Figure 2 illustrates the process of
shattering a loop-containing Morse crystal into a small collection of shattered Morse
crystals. We remark that we choose to use the Reeb graph to again trade off fidelity
with efficiency – Computing the Reeb graph of a crystal takes near-linear time
whereas computing a set of generating 1-cycles of an input crystal takes time cubic
in the size of the crystal. (Note that both methods require a 2-skeleton of the crystal
as input; in extreme cases this takes time cubic in the size of the crystal to compute.)
Furthermore, as we will show below, the use of the Reeb graph also provides a natu-
ral and simple way to further segment the input crystal; while in general, subdividing
a domain to remove all non-trivial 1-cycles does not appear to be an easy problem.

3.3.1 The Reeb Graph

Let $f : X \to \mathbf{R}$ be a continuous function. A *level set* of f with value $\alpha \in \mathbf{R}$ is
the set $f^{-1}(\alpha) = \{x \in X : f(x) = \alpha\}$. Two points $x, y \in X$ are *equivalent*
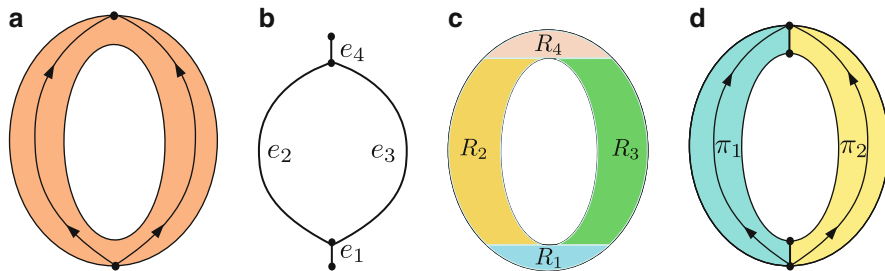
Fig. 2 (**a**) Example of a Morse crystal which contains a large loop. For this example, the height
function is used. (**b**) The Reeb graph of this crystal. (**c**) Regions of the crystal corresponding to
the four edges of its Reeb graph. (**d**) The crystal is partitioned into two shattered Morse crystals
according to which regions of (**c**) each integral curve passes through. Here, the integral curves
of the left shattered crystal pass through regions $\{R_1, R_2, R_4\}$, and integral curves of the right
shattered crystal pass through regions $\{R_1, R_3, R_4\}$

with respect to relation R, denoted xRy, if x and y reside in the same connected component of some level set. The *Reeb graph of* f [29], denoted $\mathscr{R}_f(X)$, is the quotient space X_R. Intuitively, the Reeb graph captures the merging and splitting behavior of the level sets of f. $\mathscr{R}_f(X)$ has a graph structure, and its *nodes* refers to those points such that either their up-degree or their down-degree is not 1 (where "up" and "down" refer to order in function values). An *arc* of the Reeb graph is a maximal connected component after removing nodes from $\mathscr{R}_f(X)$. See Fig. 2b for an example, where the Reeb graph of the height function defined on a torus has four nodes and four arcs.

The Reeb graph depends only on the 2-skeleton of its domain. Thus, we must extract this information for each crystal. The Rips complex provides a 2-skeleton that is sufficient for extracting topological information. It is readily computed from G_{knn} by removing edges whose endpoints lie in disparate crystals, and then transforming each 3-clique among the remaining edges into a 2-simplex. The fast randomized algorithm of Harvey et al. [21] is then used to produce the Reeb graph of each crystal.

3.3.2 Shattered Morse Crystal Decomposition

Let $G_C = (V_C, E_C)$ denote the k-nearest neighbor graph of X constrained to a Morse crystal $C \subseteq X$ output by [16]. Let $\mathscr{R}(C) = (V_R, E_R)$ be the Reeb graph of C. Let g map each edge of G_C to its set of corresponding Reeb graph edges. Given a discrete integral curve π, $g(\pi)$ denotes its image (which is also a path) in the Reeb graph $\mathscr{R}(C)$. Two points $x, y \in X$ are *shattered Morse crystal equivalent*, denoted $x \widehat{\sim} y$, if $g[\pi(x)] = g[\pi(y)]$. That is, two points are equivalent if they are in the same original crystal output by [16], and the discrete integral curves passing through them map to the same set of Reeb graph edges. The resulting quotient space $X_{\widehat{\sim}}$ is the *shattered Morse crystal decomposition of f*.

Topological Simplification   In practice, $\mathscr{R}(C)$ may contain a variety of low-persistence topological features (topological noise) that can result in an oversegmentation of the crystal. Fortunately, features arising from spurious bumps and small loops can be eliminated with the help of the *extended persistence* algorithm [1, 11] Specifically, the lowest and highest points of each loop in $\mathscr{R}(C)$ are paired using the algorithm, and if their difference in function value is below a user-specified threshold υ, the loop can be removed from $\mathscr{R}(C)$.

Recall that the input crystal C itself may be obtained after some topological simplification as described in Sect. 3.2. Hence it may contain multiple extreme points inside, and an integral path π in C may end up at some local min and max, instead of the global min and max. Now after $\mathscr{R}(C)$ is topologically simplified into $\mathscr{R}'$, it may have fewer extrema than the input crystal C. This means that the image of an integral path π in $\mathscr{R}'$ may start and/or end with some interior points, instead of with extreme points in $\mathscr{R}'$. This causes some technical problem when comparing whether two integral paths are equivalent or not in the simplified Reeb graph.

To reconcile this disparity, the notion of steepest ascending (x^+) and descending (x^-) neighbors of a point $x \in C$ is modified to take into account the structure of $\mathscr{R}'$. Specifically, if x is a local maximum in C but not in $\mathscr{R}'$, then its steepest ascending neighbor is defined to be its upper neighbor in the simplified Reeb graph $\mathscr{R}'$. An analogous redefinition is used for the steepest descending neighbors of local minima.

Computation Given a Morse crystal C output from [16], we first compute its Reeb graph $\mathscr{R}(C)$ using the algorithm from [20], and simplify it to $\mathscr{R}'$. Next, for two points $x, y \in C$, to test for their membership, we need to first compute their corresponding integral paths in C, and then check for their images.

To compute their corresponding integral paths $\pi(x)$ and $\pi(y)$, we use the same method as described in Sect. 3.2 but from the modified forests $T^+ = (C, E^+)$ and $T^- = (C, E^-)$ with $E^+ = \{(x^+, x) : x^+ \neq x\}$ and $E^- = \{(x^-, x) : x^- \neq x\}$. These forests encode the steepest ascending and descending paths of a point in C based on the modified definitions of steepest neighbors as introduced above. Next, we need to compute the images $g[\pi(x)]$ and $g[\pi(y)]$. For simplicity, in our current implementation, we compute $g[\pi(x)]$ as the sequence of Reeb graph arcs stabbed by the images of vertices from $\pi(x)$ (instead of as the concatenation of the images of edges from $\pi(x)$).[1] Two points x and y are equivalent if their integral paths produce the same sequence of Reeb graph arcs in $\mathscr{R}'$.

4 Results

We begin by investigating a pair of synthetic low-dimensional point cloud datasets. The underlying scalar functions of these datasets are precisely known and they exhibit simple structure, yet their Morse crystal decompositions contain crystals with loops. We show that the shattered Morse crystal decompositions of these datasets successfully resolve the loops. To conclude, we examine a real dataset consisting of points sampled from a scalar function whose domain is a 5-manifold embedded in $\mathbf{R}^9$. Our method successfully identifies and corrects problematic crystals in this dataset.

4.1 Torus

As a simple example, we illustrate a dataset consisting of 1,000 points sampled randomly from a 2-torus of unit diameter, with the scalar value at each point assigned

---

[1]We remark that such sequences produce the same equivalence relations as the one defined earlier when the input points are dense enough so that there is no long edge whose image spans several Reeb graph arcs.
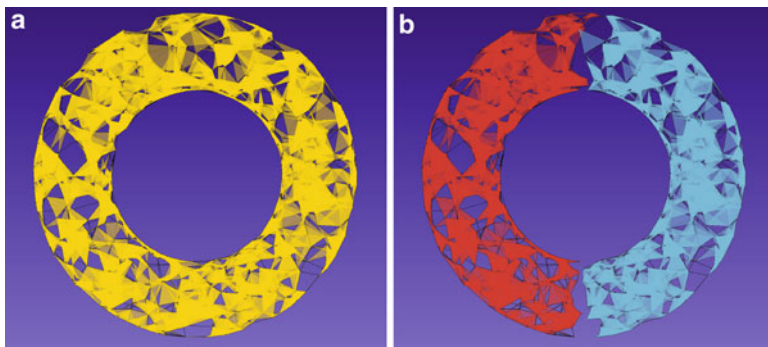
Fig. 3 Simple example of the shattered Morse crystal decomposition of a domain containing a loop. (**a**) The Morse crystal decomposition of this dataset results in a single crystal containing a large loop. (**b**) The large crystal is shattered into two parts to eliminate the loop

according to the height function (see Fig. 3). To approximate the underlying domain, $k = 20$ is used, and a threshold of $\tau = 0.1$ is used for topological simplification. The (trivial) Morse crystal decomposition is shown in Fig. 3a, i.e., the entire domain is assigned to a single Morse crystal containing a large loop. Figure 3b shows the shattered Morse crystal decomposition, which partitions the domain into two regions, neither of which contains the loop of the torus.

4.2 Mixture of Gaussians

The torus example of Sect. 4.1 exhibits a loop that is attributable to the topology of the domain. However, the algorithm in [16] can produce crystals containing loops even for scalar functions defined on simply connected domains. To demonstrate this phenomenon, we construct a simple example by taking a mixture of five 2D Gaussians (see Fig. 4). This function contains a total of eight crystals, half of which contain large loops. The shattered Morse crystal decomposition correctly shatters the crystals with loops, yielding a final segmentation of the domain into twelve regions. Naturally, the Morse crystals which do not contain loops remain intact during the shattering process.

4.3 Optimization Dataset

In this experiment we demonstrate the applicability of our algorithm to the analysis of the structure of an optimization problem. Given two images with image point correspondences, the goal is to estimate the translation and rotation of two calibrated cameras. This problem can be formulated as a minimization of the
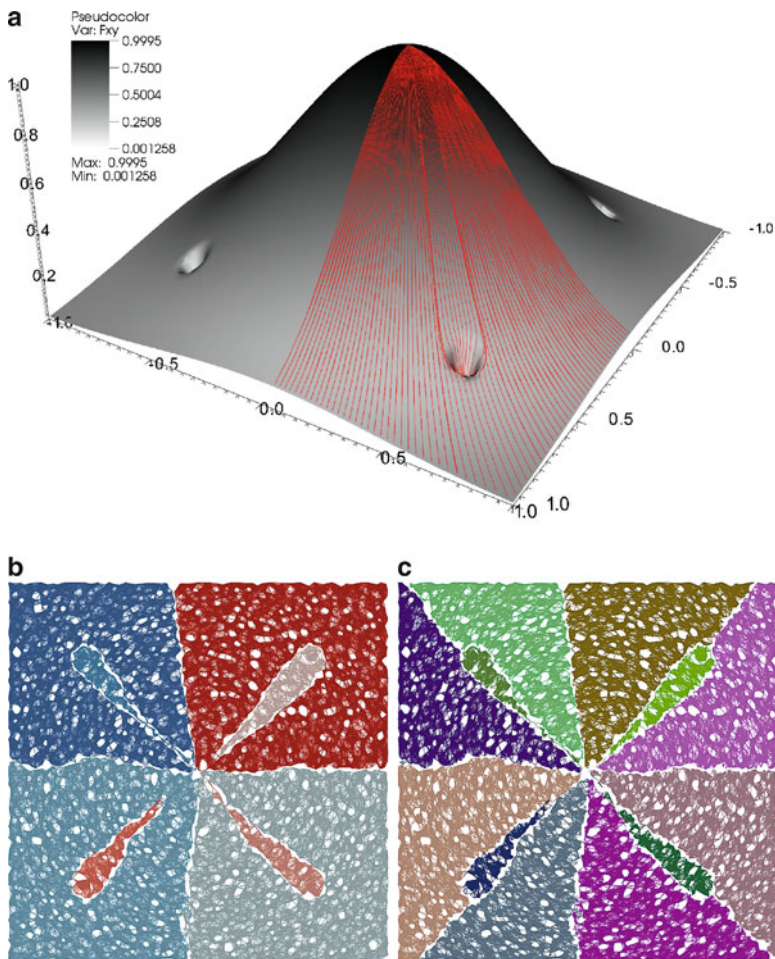
Fig. 4 Example of loops arising in Morse crystals even when the domain is simply connected. (**a**) Mixture of five Gaussians. Some discrete integral curves are shown in *red*. (**b**) The Morse crystal decomposition results in eight crystals, half of which contain loops. (**c**) The shattered Morse crystal decomposition successfully resolves the four loop-harboring crystals

algebraic error [19] which – given two corresponding points $x = [x_1 \ x_2 \ 1]^T$ and $x' = [x'_1 \ x'_2 \ 1]^T$ on the image plane in the respective coordinate frames of the two cameras – can be defined as:

$$x^T E x' = 0. \tag{1}$$

$E = [t]_\times R$ is a 3×3 rank-2 matrix, called the *essential matrix*, with the unit vector t describing the relative position, or translation, between the two cameras, and the orthogonal rotation matrix R representing the relative camera orientation. Both t and R are expressed in the coordinate frame of x. In practice, (1) can usually not be

satisfied exactly due to the presence of noise in x and x'. The problem is, therefore, formulated as an optimization of the sum of squared residuals

$$f(R, t) = f(E) = \sum_i (x_i^T E x_i')^2 \qquad (2)$$

over all point pairs $< x_i, x_i' >$. In order to minimize f, one needs to determine the nine elements of E. For more detailed information on the formulation of this problem and how it is solved in practice see the manuscript by Lindstrom and Duchaineau [24].

In order to understand the structure of f we computed 5,000 random samples of f in 9-dimensional parameter space. Due to the formulation of the problem we know that E only has 5 degrees of freedom [24], three to describe the rotation and two to determine the translation up to scale. f, hence, defines a 5-dimensional manifold embedded in 9-dimensional space.

Figure 5 shows the Morse crystal decomposition of the 5,000 point samples using 17 nearest neighbors. For these images, the 9-dimensional points have been projected onto their first two principal components. Figure 5b reveals a Morse crystal in this dataset whose Reeb graph contains two prominent loops (a topological double-torus). By applying the proposed Reeb graph shattering technique, the crystal is split into subregions.

4.4 Time Complexity

Since the shattered Morse crystal decomposition requires the 2-skeletons of the Morse crystals, it can become expensive to compute if k is chosen to be large (requiring, in the worst case, time cubic in the number of vertices). However, choosing a large k is not necessary in practice since G_{knn} is only meant to capture
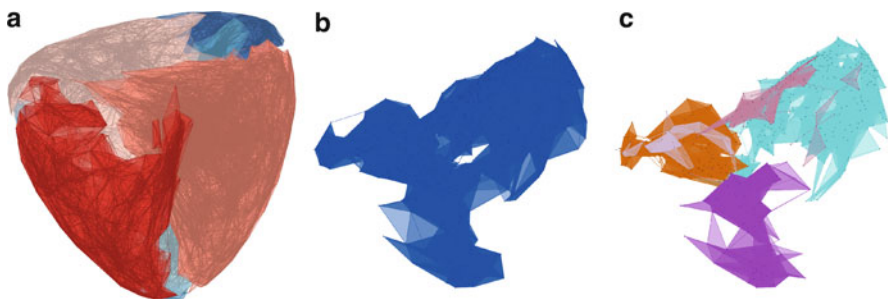
Fig. 5 Optimization dataset visualized by projecting the 9-dimensional points onto their first two principal components. (a) The Morse crystal decomposition consisting of six crystals. (b) A Morse crystal whose Reeb graph contains two prominent loops is partitioned into a small collection of shattered Morse crystals (c)

the topology *locally* around each point. In our experience, combining a reasonably small k (e.g., $10 < k < 40$) with topological simplification techniques works well for capturing the important topological features of the dataset at hand while minimizing the prohibitive time complexity of 2-skeleton construction.

5   Conclusions

Reliable estimation of topological structures of high-dimensional functions is essential for accurate topology-based visualization and analysis. Feature-based visualizations and statistical analyses of high-dimensional functions based on the analysis of Morse crystals commonly rely on the notion that a Morse crystal is monotone and has genus 0. Morse crystals with a common source (maximum) and sink (minimum), however, are not separated correctly by the approximate Morse-Smale complex algorithm, which can lead to the creation of Morse crystals with a genus ≥ 1.

We described a simple and novel algorithm to augment the previous Morse crystal decomposition algorithm to detect and correct certain falsely merged Morse crystals containing possibly multiple loops. Using the per-crystal Reeb graph we detect crystals containing loops and shatter them into multiple, loop-free crystals.

We illustrated the problem of crystals containing loops using two analytic examples and showed that our algorithm produces a correct complex in both cases. We used our algorithm to analyze the structure of a real-world optimization problem, demonstrating its practical relevance. Our algorithm produces a more accurate approximation of the Morse-Smale complex in high-dimensions.

While the proposed algorithm is an improvement, it is still lacking in some areas. Specifically, it is only capable of detecting and resolving those loops which are evident from the structure of the Reeb graph. Thus, it may fail to detect some loops in the domain which do not induce loops in the Reeb graph. Additionally, like its predecessor, it is incapable of dealing with any cycles of dimension 2 or greater that may be present in the dataset. Additional research in topology-sensitive clustering methods will strive to provide insight into how to grapple with these difficulties.

Computation of topological structures of point-cloud data rely on proper estimation of the structure of the domain. The study and development of methods for constructing optimal neighborhood/proximity graphs, hence, promises to enable more accurate approximations of topological structures. In future we also plan to investigate further applications of the approximate Morse-Smale complex to improve understanding of the structure of high-dimensional scalar functions.

References

1. Agarwal, P.K., Edelsbrunner, H., Harer, J., Wang, Y.: Extreme elevation on a 2-manifold. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, SCG 04, Brooklyn, New York, pp. 357–365. ACM, New York (2004)
2. Aurenhammer, F.: Voronoi diagrams – a survey of a fundamental geometric data structure. ACM Comput. Surv. **23**(3), 345–405 (1991)
3. Beucher, S.: Watersheds of functions and picture segmentation. In: Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82, vol. 7, pp. 1928–1931 (1982)
4. Beucher, S., Lantuejoul, C.: Use of watersheds in contour detection. In: International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France (1979)
5. Bremer, P.T., Weber, G., Pascucci, V., Day, M., Bell, J.: Analyzing and tracking burning structures in lean premixed hydrogen flames. IEEE Trans. Visual. Comput. Graph. **16**(2), 248–260 (2010)
6. Carr, H., Snoeyink, J., Axen, U.: Computing contour trees in all dimensions. Comput. Geom. Theor. Appl. **24**(3), 75–94 (2003)
7. Carr, H., Snoeyink, J., van de Panne, M.: Simplifying flexible isosurfaces using local geometric measures. In: IEEE Visualization '04, pp. 497–504. IEEE Computer Society, MD (2004)
8. Chazal, F., Guibas, L., Oudot, S., Skraba, P.: Analysis of scalar fields over point cloud data. In: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '09, New York, pp. 1021–1030. Society for Industrial and Applied Mathematics, Philadelphia (2009)
9. Chazal, F., Guibas, L., Oudot, S., Skraba, P.: Persistence-based clustering in riemannian manifolds. Tech. Rep. RR-6968, INRIA (2009)
10. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. **17**(8), 790–799 (1995)
11. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Extending persistence using poincaré and lefschetz duality. Found. Comput. Math. **9**(1), 133–134 (2009)
12. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE TPAMI **24**, 603–619 (2002)
13. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. Discrete Comput. Geom. **28**, 511–533 (2002)
14. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. Discrete Comput. Geom. **30**, 87–107 (2003)
15. Edelsbrunner, H., Harer, J., Natarajan, V., Pascucci, V.: Morse-Smale complexes for piecewise linear 3-manifolds. In: Proceedings of the Nineteenth Annual Symposium on Computational Geometry, San Diego, CA, pp. 361–370. ACM, New York (2003)
16. Gerber, S., Bremer, P.T., Pascucci, V., Whitaker, R.: Visual exploration of high dimensional scalar functions. IEEE Trans. Visual. Comput. Graph. **16**(6), 1271–1280 (2010)
17. Gyulassy, A., Natarajan, V., Pascucci, V., Hamann, B.: Efficient computation of Morse-Smale complexes for three-dimensional scalar functions. IEEE Trans. Visual. Comput. Graph. **13**(6), 1440–1447 (2007)
18. Gyulassy, A., Duchaineau, M., Natarajan, V., Pascucci, V., Bringa, E., Higginbotham, A., Hamann, B.: Topologically clean distance fields. IEEE Trans. Visual. Comput. Graph. **13**(6), 1432–1439 (2007)
19. Hartley, R., Zisserman, A.: Multiple View Geometry, 2nd edn. Cambridge University Press, London (2003)
20. Harvey, W., Wang, Y.: Generating and exploring a collection of topological landscapes for visualization of scalar-valued functions. Comput. Graph. Forum **29**(3), 9931002 (2010)
21. Harvey, W., Wang, Y., Wenger, R.: A randomized O(m log m) time algorithm for computing Reeb graphs of arbitrary simplicial complexes. In: Proc. Annual Symp. on Computational Geometry 2010, SoCG '10, pp. 267–276. ACM, New York, NY, USA (2010)

22. Jaromczyk, J.W., Abstract, G.T.T.: Relative neighborhood graphs and their relatives. Proc. IEEE **80**(9), 1502–1517 (1992)
23. Laney, D., Bremer, P.T., Mascarenhas, A., Miller, P., Pascucci, V.: Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. IEEE Trans. Visual. Comput. Graph. **12**(5), 1052–1060 (2006)
24. Lindstrom, P., Duchaineau, M.: Factoring algebraic error for relative pose estimation. Tech. Rep. LLNL-TR-411194, Lawrence Livermore National Laboratory (2009)
25. Morse, M.: Relations between the critical points of a real functions of n independent variables. Trans. Am. Math. Soc. **27**, 345–396 (1925)
26. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J. ACM **45**(6), 891–923 (1998)
27. Najman, L., Schmitta, M.: Watershed of a continuous function. Signal Process. Mathematical Morphology and its Applications to Signal Processing **38**(1), 99–112 (1994)
28. Pascucci, V., Scorzelli, G., Bremer, P.T., Mascarenhas, A.: Robust on-line computation of Reeb graphs: Simplicity and speed. ACM Trans. Graph. **26**(3), 58 (2007)
29. Reeb, G.: Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [on the singular points of a complete integral pfaff form or of a numerical function]. Comptes Rendus Acad. Science Paris **222**, 847–849 (1946)
30. Sheikh, Y., Kahn, E., Kanade, T.: Mode-seeking by medoidshifts. In: IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007, pp. 1–8 (2007)
31. Tarjan, R.E., van Leeuwen, J.: Worst-case analysis of set union algorithms. J. ACM **31**, 245–281 (1984)
32. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: ECCV (4), pp. 705–718 (2008)
33. Vietoris, L.: Über den höheren Zusammenhang kompakter Räume und eine Klasse von zusammenhangstreuen Abbildungen. Math. Ann. **97**(1), 454472 (1927)
34. Vincent, L., Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE Trans. Pattern Anal. Mach. Intell. **13**(6), 583–598 (1991)
35. Zhu, X., Sarkar, R., Gao, J.: Shape segmentation and applications in sensor networks. In: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pp. 1838–1846 (2007)

Efficient Computation of Persistent Homology for Cubical Data

Hubert Wagner, Chao Chen, and Erald Vuçini

1 Introduction

Persistent homology [10, 11] has drawn much attention in visualization and data analysis, mainly due to the fact that it extracts topological information that is resilient to noise. This is especially important in application areas, where data typically comes from measurements which are inherently inexact. Although direct application of persistent homology is still at an early stage, closely related concepts like size functions [3], contour trees [1, 4], Reeb graphs [24] and Morse-Smale complexes [14] have been successfully used.

The under-usage of persistence in applications is largely due to its high computational cost. The standard algorithm [10] takes cubic running time, which can be prohibitive even for small size data (e.g., $64 \times 64 \times 64$). In addition to the high time complexity, there are two further issues: (1) the memory consumption of the currently available implementations, even for small data sizes, is very large and hence prohibitive for commodity computers, and (2) the focus of several applications is in

H. Wagner (✉)
Jagiellonian University, 31-007 Kraków, Poland

Vienna University of Technology, Karlsplatz 13, 1040 Vienna, Austria
e-mail: hubert.wagner@ii.uj.edu.pl

C. Chen
Institute of Science and Technology, Am Campus 1, 3400 Klosterneuburg, Austria

Vienna University of Technology, Karlsplatz 13, 1040 Vienna, Austria
e-mail: chao.chen@ist.ac.at

E. Vuçini
VRVis Center for Virtual Reality and Visualization Research-Ltd, Donau-City-Strasse 1, 1220 Vienna, Austria

Vienna University of Technology, Karlsplatz 13, 1040 Vienna, Austria
e-mail: erald.vucini@vrvis.at

data of higher dimensions, e.g., 4D, 5D or higher. Few implementations for general dimension are available and the existing ones do not scale well with the increase of dimensions, hence introducing larger computation times and memory inefficiency.

In this paper, we present an efficient framework that computes persistent homology exactly.[1] To our knowledge, this is the very first implementation that can handle large and high dimensional data in reasonable time and memory. We focus on uniformly/regularly sampled data which is common in visualization and data analysis, i.e., image data consisting of pixels (2D images), voxels (3D scans, simulations), or their higher-dimensional analogs, e.g., 4D time-varying data. In this work, we use the name "cubical" for such data.

We depart from the standard method which involves triangulating the space, and computing persistent homology of the resulting simplicial complex [10, 11]. We use cubical complexes [15], which do not require subdivision of the input. The advantage is twofold. First, the size of the complexes is significantly reduced, especially for high dimensional data (see Sect. 5 for a quantitative analysis). Second, cubical complexes allow for the usage of more compact data-structures.

The standard persistence algorithm requires the computation of a sorted boundary matrix. This step can be a significant bottleneck, especially in terms of memory consumption. In this work we provide an efficient and compact algorithm for this step, using techniques from (non-persistence) cubical homology [15] (see Sect. 4).

In Sect. 7, we present experimental results. Comparison with existing packages shows significant efficiency improvement. We also explore how our method scales with respect to data size and dimension. In conclusion, our framework can handle data of large size and high dimension, and therefore, makes the persistence computation of cubical data more feasible.

2   Related Work

The first algorithm for computing persistence [11] has cubic running time with respect to the complex size (which is larger than the input size). Morozov [20] formulated a worst case scenario for which the persistence algorithm reaches this asymptotic bound. When focusing on 0-dimensional homology, union-find data structures can be used to compute persistence in time $O(n\alpha(n))$ [10], where α is the inverse of the Ackermann functions and n is the input size. Milosavljevic et al. [18] compute persistent homology in matrix multiplication time $O(n^\omega)$ where the currently best estimation of ω is 2.376. Chen and Kerber [5] proposed a randomized algorithm whose complexity depends on the number of persistence pairs with persistence above a certain threshold. Despite showing better theoretical complexity,

---

[1] We emphasize that our work focuses on computing persistence exactly. There are approximation methods which trade accuracy for efficiency. See Sect. 2.

it is unclear whether these methods are better than the standard persistence algorithm in practice.

In terms of implementation, Morozov [19] provides a C++ code for the standard persistence algorithm. Chen and Kerber [6] devised a technique which, in practice, significantly improves the matrix-reduction part of this algorithm. We build upon their work, to improve the overall performance of the persistence algorithm.

The application of cubical homology is straightforward in the areas of image processing and visualization, where cubical data is the typical input. Non-persistent cubical homology has found practical applications in a number of cases [21, 22]. A few attempts of cubical persistence computations have been made recently [16, 25]. They do not, however, tackle the problem of performance. In [25], experiments with datasets containing several thousands of voxels are reported. In comparison, real world applications require processing of data in the range of millions or billions of voxels.

Recently, Mrozek and Wanner [21] showed that cubical persistent homology can be used for medium-sized datasets. A detailed performance summary is given for 2D and 3D images. One downside of this approach is the dependency on the number of unique values of the image. When such a number is close to the input size, the complexity is prohibitively high. In Sect. 7, we compare our method with this algorithm.

We must differentiate between the two main types of persistence computations: exact and approximative (where the persistence is calculated approximately). While we focus on the first type, approximation is less computationally intensive and is therefore important for large data. Bendich et al. [2] use octrees to approximate the input. A simplicial complex of small size is then used to complete persistence computation.

3 Theoretical Background

3.1 *Simplicial and Cubical Complexes*

In computational topology, simplicial complexes are frequently used to describe topological spaces. A simplicial complex consists of simplices like vertices, edges and triangles. In general, a *d-simplex* is the convex hull of $d + 1$ points. The convex hull of any subset of these $d + 1$ points is a *face* of this d-simplex. A collection of simplices, K, is a *simplicial complex* if: (1) for any simplex in K, all its faces also belong to K, and (2) for any two simplices in K, their intersection is either empty, or a common face of them.

Next, we define cubical complexes. An *elementary interval* is defined as a unit interval $[k, k + 1]$, or a degenerate interval $[k, k]$. For a d-dimensional space, a *cube* is a product of d elementary intervals I: $\prod_{i=1}^{d} I_i$. The number of non-degenerate intervals in such a product is the *dimension* of this cube. 0-cubes, 1-cubes, 2-cubes
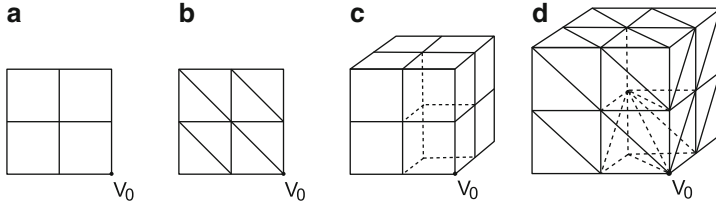
Fig. 1 Cubical complex triangulations: (**a**) a 2D cubical complex, and (**b**) its triangulation, (**c**) a 3D cubical complex, and (**d**) its triangulation (only simplices which contain V_0 are drawn)

and 3-cubes are vertices, edges, squares and 3D cubes (voxels) respectively. Given two cubes: $a, b \subseteq R^d$, a is a *face* of b if and only if $a \subseteq b$. A *cubical complex* of dimension d is a collection of cubes of dimension at most d. Similarly to the definition of a simplicial complex, it must be closed under taking faces and intersections.

In this paper, we will use cubical complexes to describe the data. In Fig. 1 we show 2-dimensional and 3-dimensional cubical complexes, describing a 2D image of size 3×3 and a 3D image of size $3 \times 3 \times 3$. The corresponding simplicial complex representations are also shown. We use one specific triangulation, namely, the *Freudenthal triangulation* [13,17], which is easy to extend to general dimension.

3.2 Boundary Matrix

For any d-dimensional *cell* (that is: simplex or cube), its *boundary* is the set of its $(d - 1)$-dimensional faces. This extends linearly to the boundary of a set of d-cells, namely, a d-*chain*. Specifically, the boundary of a set of cells is the modulo 2 sum of the boundaries of each of its elements. In general, if we specify a unique index for each simplex, a d-chain corresponds to a vector in $\mathbb{Z}_2^{n_d}$, where n_d is the number of d-dimensional cells in the complex. Furthermore, the d-dimensional boundary operator can be written as a $n_{d-1} \times n_d$ binary matrix whose columns are the boundaries of d-cells, while rows contain $(d - 1)$-cells.

3.3 Persistent Homology

We review persistent homology [10, 11], focusing on $\mathbb{Z}_2$ homology. Due to space limitation, we do not introduce homology in this paper. Please see [12] for an intuitive explanation, and [10, 23] for related textbooks.

Given a topological space $\mathbb{X}$ and a *filtering function* $f : \mathbb{X} \rightarrow \mathbb{R}$, *persistent homology* studies homological changes of the sublevel sets, $\mathbb{X}^t = f^{-1}(-\infty, t]$. The algorithm captures the birth and death times of homology classes of the sublevel

set as it grows from $\mathbb{X}^{-\infty}$ to $\mathbb{X}^{+\infty}$, e.g., components as 0-dimensional homology classes, tunnels as 1-dimensional classes, voids as 2-dimensional classes, and so on. By birth, we mean that a homology class comes into being; by death, we mean it either becomes trivial or becomes identical to some other class born earlier. The persistence, or lifetime of a class, is the difference between the death and birth times. Homology classes with larger persistence reveal information about the global structure of the space $\mathbb{X}$, as described by the function f.

Persistence can be visualized in different ways. One well-accepted idea is the persistence diagram [7], which is a set of points in a two-dimensional plane, each corresponding to a persistent homology class. The coordinates of such a point are the birth and death time of the related class.

An important justification of the usage of persistence is the stability theorem. Cohen-Steiner et al. [7] proved that for any two filtering functions f and g, the difference of their persistence is always upperbounded by the L^∞ norm of their difference:

$$\|f - g\|_\infty := \max_{x \in \mathbb{X}} |f(x) - g(x)|.$$

This guarantees that persistence can be used as a signature. Whenever two persistence outputs are different, we know that the functions are definitely different.

In our framework, for 2D images we assume 4-connectivity. In general, for d-dimensional cubical data, we use $2d$-connectivity.

3.4 Persistence Computation

Edelsbrunner et al. [11] devised an algorithm to compute persistent homology, which works in cubic time (in the size of a complex). It requires preprocessing of the data (also see Fig. 2). In case of images, function f is defined on all pixels/voxels. First, these values are interpreted as values of vertices of a complex. Next, the *filtration* of the complex is computed and the *sorted boundary matrix* is generated. This matrix is the input to the *reduction algorithm*.

Filtration can be described as adding cells with increasing values to a complex, one by one. To achieve this, a *filtration-building algorithm* extends the function to all cells of the complex, by assigning each cell the maximum value of its vertices. Then, all cells are sorted in ascending order according to the function value, so that
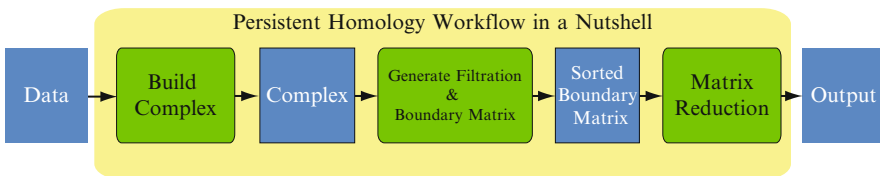

Fig. 2 A workflow of the persistent homology computation

each cell is added to the filtration after all of its faces. Such a sequence of cells is called a *lower-star filtration*. Having calculated the ordering of cells, a sorted boundary matrix can be generated.

In the reduction step, the algorithm performs column reductions on the sorted boundary matrix from left to right. Each new column is reduced by addition with the already reduced columns, until its lowest nonzero entry is as high as possible. The reduced matrix encodes all the persistent homology information.

4 Efficient Filtration-Building Algorithm

The filtration-building is one of the main bottlenecks of the persistence algorithm. A straightforward approach would choose to store the boundary relationship between cells and their faces. In this section, we describe the first major contribution of the paper, a new algorithm for the filtration-building step. Our algorithm uses the regular structure of cubical complex and adapts a compact data structure which has shown its utility in non-persistent cubical homology.

### 4.1 Cubical Complex Representation

We first describe CubeMap, a compact representation of cubical complexes. To the best of our knowledge, a similar structure was first introduced in CAPD library [8] for non-persistent cubical homology.

For an example 2D image with 5×5 pixels see Fig. 3. Due to the regular structure, relationship between cells can be read immediately from their coordinates. We can store the necessary information (i.e. order in the filtration, function value) for each cell in a 9×9 array (Fig. 3c). We can immediately get the dimension of any cell (whether it is a vertex, edge, or square), as well as its faces and *cofaces*, namely, cells of whom it is a face. We do this by checking coordinates modulo 2. To explain this fact, we recall that we defined cubes as products of intervals. Even coordinates correspond to degenerate intervals of a cube.

The aforementioned properties generalize for arbitrary dimensions. This is due to the inductive construction of cubical complexes, and is related to cubes being products of intervals.

Let us consider input data of dimension d and size w^d, where w is the number of vertices in each dimension. We store information attached to cells in a d-dimensional array with $(2w-1)^d$ elements. This array is composed of overlapping copies of arrays of size 3^d. We call this structure the $CubeMap$.

The major advantage of the proposed data-structure is the improved memory efficiency. Boundary relations are implicitly encoded in the coordinates of cells. The coordinates themselves are also implicit. Furthermore, we can randomly access
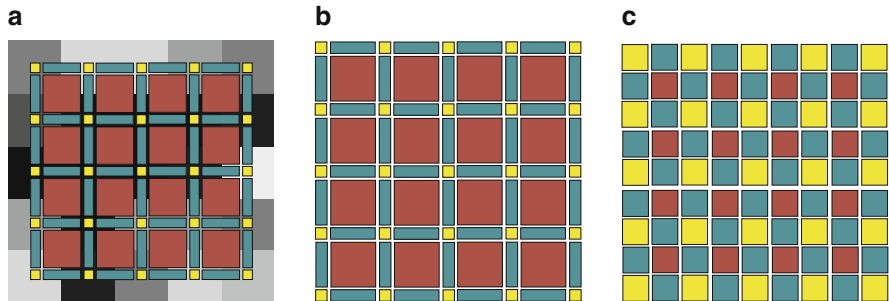
Fig. 3 (**a**) Cubical complex built over a gray-scale 2D image with 5 × 5 pixels. Each vertex (*yellow*) corresponds to a pixel. Edges (*blue*) and cubes (*red*) are constructed accordingly. (**b**) The cubical complex itself. (**c**) The corresponding CubeMap, all informations for filtration-building are encoded in a 9 × 9 array. Each element corresponds to a single cube
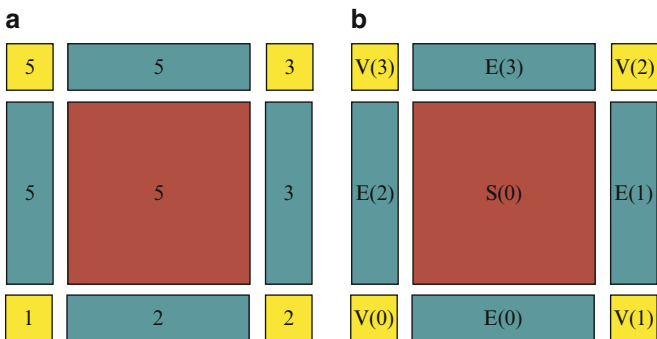
Fig. 4 (**a**) Values of f assigned to vertices and extended to all cubes. (**b**) Cells are assigned indices in the filtration. These indices are separate for each dimension. Vertices are marked as V, edges as E, squares as S

each cell and quickly locate its boundaries. See Sect. 6 for further details and Sect. 7 for an experimental justification.

4.2 Filtration-Building

Let us now present an efficient algorithm to compute a filtration of a cubical complex induced by a given function f (see Algorithm 1). We use the CubeMap datastructure to store additional information for each cube (function value, filtration order). The outcome of this algorithm is a sorted boundary matrix, being the input of the reduction step. Since in case of cubical data boundary matrices have only $O(d)$ non-zero elements per column, sparse representations are typically used.

The intuition behind the algorithm is that when we iterate through all vertices in *descending* order, we know that the vertices' cofaces, which were not added to the filtration, belong to their lower-stars, and can be added to the filtration. We cannot build the boundary matrix in the same step, since the indices of the adjacent cells might be not yet computed. Do note that on line 5, filtration indices are assigned from higher to lower. Figure 4 illustrates the algorithm. Exploiting the properties of cubical complexes makes this algorithm efficient (refer to Sect. 6 for details).

Algorithm 1: Computing filtration and sorted boundary matrix

Input: function f, given on vertices of a cubical complex K
Output: sorted boundary matrix, extension of function f to all cubes of K
1: sort vertices of K by values of f (descending)
2: **for** each vertex V_i in sorted order **do**
3:     **for** each cube C_j with V_i as one of its vertices **do**
4:         **if** C_j was **not** assigned filtration index **then**
5:             assign next (smaller) filtration index to C_j
6:             $f(C_j) \leftarrow f(V_i)$.
7: **for** each cube C_i of K **do**
8:     column $\leftarrow$ filtration index of C_i
9:     **for** each cube B_j in boundary of C_i **do**
10:         row $\leftarrow$ filtration index of B_j
11:         boundary matrix(row, column) $\leftarrow 1$

---

5   Sizes of Complexes

When switching from simplicial complexes to cubical complexes, the size of the complex is significantly reduced. This is a clear improvement in both memory and runtime efficiency. We should emphasize that the complexity of the standard reduction algorithm is given in the size of the complex, not the number of vertices. Therefore, reducing the size of a complex has a significant impact.

In this section, we analyze how the ratio of the sizes of simplicial and cubical complexes increases with respect to the data dimension. We show that this ratio increases exponentially with the dimension, which motivates the usage of cubical approaches, such as ours. For simplicity we disregard boundary effects, assuming that the number of cells lying on the boundary is insignificant.[2]

In Fig. 1, we show examples of cubical complexes and their triangulations. The ratio between the number of cofaces of the vertex V_0 in a simplicial and in a cubical complex is (6 : 4) and (26 : 8) for 2D and 3D complexes, respectively. This is also the ratio of the size of simplicial and cubical complexes, since these selected cells *generate* the whole complex.

---

[2]This assumption is realistic when complexes are large.

Table 1 Lower-bounds of the size ratios ρ_d

	Dimension (d)	1	2	3	4	5	6	7
Optimal	τ_d	1	2	5	16	67	308	1,493
	Lower-bound of ρ_d	1.0	1.5	2.75	5.625	12.937	33.968	90.265
Freudenthal	τ_d	1	2	6	24	120	720	5,040
	Lower-bound of ρ_d	1.0	1.5	3.0	7.125	19.375	60.156	213.062

For a d-dimensional data, we denote the concerned ratio as $\rho_d = S_d/C_d$, where C_d and S_d are the sizes of a cubical complex and its triangulation, respectively. It is nontrivial to give an exact formula of ρ_d, since the minimal-cardinality cube-triangulation is an open problem [26]. Here we give a lower-bound of ρ_d for $d \leq 7$ by triangulating all cubes of a cubical complex separately in each dimension. When triangulating a d-cube, we count only the resulting d-simplices, and their $(d-1)$-dimensional intersections. Finally, taking into account the fact that certain simplices will be common faces of multiple higher-dimensional simplices, we get

$$\rho_d \geq \frac{\sum_{i=0}^{d} \binom{d}{i}\tau_i + \sum_{i=0}^{d-1} \binom{d}{i+1}(\tau_{i+1} - 1)}{2^d}$$

where τ_d is the number of d-simplices in a triangulation of a d-cube.

In Table 1 we present the values of such lower-bounds for different dimensions ($d = 1, \cdots, 7$). We consider two cases: optimal triangulation [26] and Freudenthal, using $d!$ simplices. It is clear that in both cases the lower-bound increases exponentially with respect to the data dimension.

This observation leads to the following conjecture. This conjecture, if correct, shows how algorithms based on cubical and simplicial complexes scale with respect to the dimension.

Conjecture 1. ρ_d increases exponentially in d.

6   Implementation Details

In this section we briefly comment on the techniques we used to enhance the performance of our implementation. We focus on the choice of proper data-structures, and exploiting various features of cubical complexes. Our method is implemented in C++.

6.1   Filtration-Building Algorithm

We use a 2-pass modification of the standard filtration-building algorithm. Reversing the iteration order over the vertices does not affect the asymptotic complexity, but simplifies the first pass of the algorithm, which resulted in better performance.

We calculate the time complexity of this algorithm. To do this precisely, we assume that the dimension d is not a constant. This is a fair assumption since we consider general dimensions. We use a d-dimensional array to store our data, so random access is not $O(1)$, but $O(d)$, as it takes $d-1$ multiplications and additions to calculate the address in memory.

Let n be the size of input (the number of vertices in our complex). In total there are $O(2^d n)$ cubes in the complex. We ignore what happens at boundaries of the complex. Each d-cube has exactly $2d$ boundary cubes, and each vertex has $3^d - 1$ cofaces. Accessing each of them costs $O(d)$. This yields the following complexity of calculating the filtration and the boundary matrices: $O(d3^d n + d^2 2^d n)$.

Using the properties of CubeMap, we can reduce this complexity. Since the structure of the whole complex is regular, we can precalculate memory-offsets from cubes of different dimensions and orientations to its cofaces and boundaries. Accessing all boundary cubes and cofaces takes constant amortized time. The preprocessing time does not depend on input size and takes only $O(d^2 3^d)$ time and memory. With the CubeMap data structure, our algorithm can be implemented in $\Theta(3^d n + d2^d n)$ time and $\Theta(d2^d n)$ memory.

6.2 Storing Boundary Matrices

Now we present a suggestion regarding performance, namely, the usage of a proper data-structure for storing the columns of (sparse) boundary matrices. In [10] a linked-list data-structure is suggested. This seems to be a sub-optimal solution, as it has an overhead of at least one pointer per stored element. For 64-bit machines this is 8B - twice as much as the data we need to store in a typical situation (one 32-bit integer).

Using an automatically-growing array, such as std::vector available in STL is much more efficient (speed-up by a factor of at least 2). Also the memory overhead is much smaller – 16B per column (not per element as before). All the required operations have the same (amortized) complexity [9], assuming that adding an element at the back can be done in constant amortized time. Also, iterating the array from left to right is fast, due to memory-locality, which is not the case for linked-list implementations.

7 Results

The testing platform of our experiments is a six-core AMD Opteron(tm) processor 2.4GHz with 512KB L2 cache per core, and 66GB of RAM, running Linux. Our algorithm runs on a single core. We use 3D and 4D (3D+time) cubical data for testing and comparing our algorithm. We compare our method with existing implementations. We measure memory usage, filtration-building and reduction times.

Table 2 Memory consumption for the computation of persistence of the Aneurysm dataset for different implementations. Several down-sampled version of the original dataset were used. For specific cases the results are not reported due to memory or time limitations

	$50 \times 50 \times 50$	$100 \times 100 \times 100$	$150 \times 150 \times 150$	$200 \times 200 \times 200$	$256 \times 256 \times 256$
CAPD	500 MB	2,700 MB	16,000 MB	–	–
Dionysus	200 MB	6,127 MB	21,927 MB	49,259 MB	–
SimpPers	352 MB	3,129 MB	11,849 MB	25,232 MB	–
CubPers	42 MB	282 MB	860 MB	2,029 MB	4,250 MB

Table 3 Times (in minutes) for the computation of persistence of the Aneurysm dataset for different techniques. For SimpPers and CubPers, we report both filtration-building time and reduction time, the whole computation is the sum of the two times

	$50 \times 50 \times 50$	$100 \times 100 \times 100$	$150 \times 150 \times 150$	$200 \times 200 \times 200$	$256 \times 256 \times 256$
CAPD	0.26	12.3	134.55	–	–
Dionysus	0.32	3.03	13.74	47.23	–
SimpPers	(0.05+0.02)	(0.43+0.16)	(1.63+0.9)	(3.53+3.33)	–
CubPers	(0.01+0.001)	(0.10+0.01)	(0.33+0.13)	(0.87+0.43)	(1.25+0.78)

7.1 Comparing with Existing Implementations

We compare our implementation (referred to as CubPers) to three existing implementations:

1. **SimpPers:** (by Chen and Kerber [6]) Uses simplicial complexes. Both SimpPers and CubPers use the same reduction algorithm, but our approach uses cubical complexes and CubeMap to accelerate the filtration-building process.
2. **Dionysus:** (by Morozov [19]) This code is suited for more general complexes and computes also other information like vineyards. We adapt this implementation to operate on cubical data, by triangulating the input, which is the standard approach. Since this implementation takes a filtration as input, the time for building the filtration is not taken into account.
3. **CAPD:** (by Mrozek [21], a part of CAPD library [8]) We stress that this approach was designed for data with a small number of unique function values, which is not the case for the data we use. Additionally it produces and stores persistent homology generators which incur a significant overhead.

In Tables 2 and 3 we compare the memory and times of our approach to the aforementioned implementations. For testing we have used the Aneurysm dataset.[3] In order to explore the behavior of the algorithms when the data size increases linearly, we uniformly scale the data into 50^3, 100^3, 150^3, 200^3, using nearest neighbor interpolation. Clearly, our implementation, CubPers, outperforms other programs in terms of memory and time efficiency.

---

[3]From the Volvis repository (http://volvis.org/).

Table 4 Times (in minutes) for the computation of persistence for one million vertices in different dimensions (1–6). Both times for filtration and persistence (filtration+reduction) are given

Dimension	1D	2D	3D	4D	5D	6D
Filtration	0.017	0.05	0.15	0.55	1.65	3.70
Persistence (reduction)	0.067	0.12	0.23	0.87	4.80	17.70

Due to the usage of CubeMap, the memory usage is reduced by an order of magnitude. This is extremely important, as it enables the usage of much larger datasets on commodity computers. While SimpPers significantly improves over other methods in terms of reduction time [6], our method further improves the filtration-building time. It is also shown that using cubical complexes instead of simplicial complexes improves the reduction time.

7.2 Scalability

Table 4 shows how our implementation scales with respect to dimension. We used random data – each vertex is assigned an integer value from 0 to 1,023 (the choice was arbitrary). The distribution is uniform and the number of vertices (1,000,000) is constant for all dimensions. We can see that performance deteriorates exponentially. This is understandable, since the size of a cubical complex increases exponentially in dimension (2^d). The size of its boundary matrix increases even faster ($d\,2^d$).

In Table 5 we report the timings and memory consumptions for several 3D datasets[4] and a 4D time-varying data[5] consisting of 32 timesteps. We stress the following three observations:

- Due to the significant improvement of memory efficiency, our implementation can compute these data on commodity computers.
- Both memory and filtration-building times grow linearly in the size of data (number of voxels). This also reveals that for very large scale data ($\geq 1,000^3$), the memory consumption would be too large. In such a case, we would need an approximation algorithm (as in [2]) or an out-of-core algorithm.
- Reduction time varies for different data. Among all data-files we tested, two medium (256^3) cases (Christmas Tree and Christmas Present) took significantly more reduction time. This data-dependent behavior of the reduction algorithm is an open problem in persistent homology literature.

7.3 Understanding Time-Varying Data with Persistence

With our efficient tool, we are enabled to study 4D time-varying data using persistence. This is one of our future research focuses. We conclude this section by

---

[4]From the Volvis repository (http://volvis.org/) and ICGA repository (www.cg.tuwien.ac.at).

[5]From the Osirix repository (http://pubimage.hcuge.ch:8080/).

Table 5 Times in minutes for different 3D datasets and a 4D time-varying data (32 timesteps). Times below 0.001 min were reported as 0.00

Data set	Size	Memory (MB)	Times (min)
Silicium	$98 \times 34 \times 34$	30	(0.02+0.07)
Fuel	$64 \times 64 \times 64$	82	(0.02+0.00)
Marschner-Lobb	$64 \times 64 \times 64$	82	(0.03+0.00)
Neghip	$64 \times 64 \times 64$	82	(0.03+0.00)
Hydrogene	$128 \times 128 \times 128$	538	(0.22+0.40)
Engine	$256 \times 256 \times 128$	2,127	(1.07+0.30)
Tooth	$256 \times 256 \times 161$	2,674	(1.43+1.48)
Christmas Present	$246 \times 246 \times 221$	3,112	(2.43+264.35)
Christmas Tree	$256 \times 249 \times 256$	3,809	(3.08+11.1)
Aneurysm	$256 \times 256 \times 256$	4,250	(1.75+0.77)
Bonsai	$256 \times 256 \times 256$	4,250	(1.98+0.93)
Foot	$256 \times 256 \times 256$	4,250	(2.15+0.70)
Supine	$512 \times 512 \times 426$	26,133	(23.06+11.88)
Prone	$512 \times 512 \times 463$	28,406	(25.96+10.38)
Vertebra	$512 \times 512 \times 512$	31,415	(26.8+7.58)
Heart (4D)	$256 \times 256 \times 14 \times 32$	13,243	(20.20+1.38)

a pilot study of a dataset representing a beating heart. We treat all four dimensions of this data (3 spatial and time) equally,[6] and then compute the persistence diagrams (Fig. 5a–d). In Fig. 5e we display graphs of the Betti numbers of the sublevel sets. Blue, red, green and pink correspond to 0–3 dimensional Betti numbers, respectively.

8   Summary and Future Work

In this paper, we showed that our approach can be used to compute persistent homology for large cubical data-sets in arbitrary dimensions. Our experiments show that our method is more efficient with regard to time and memory than the existing implementations. The reduction of memory usage is especially important, as it enables the use of persistent homology for much larger datasets.

There is a wide range of directions to be considered in the future research. We consider further development of the proposed method. In particular, a parallel implementation is a promising option. Further reduction of memory usage and moving towards out-of-core computations are important directions, but also very challenging.

---

[6]In general, this may not be the right approach, as it does not assume the non-reversibility of time.
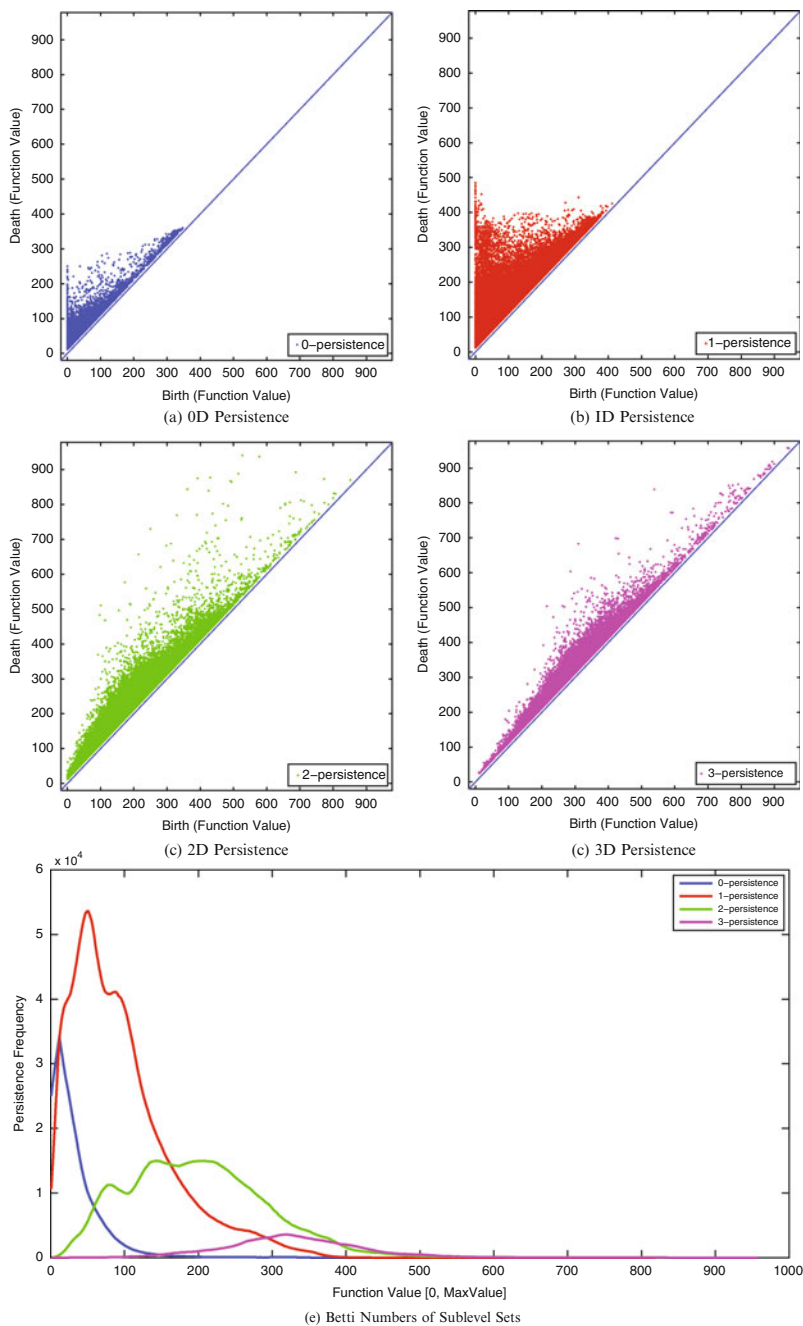
(a) 0D Persistence

(b) ID Persistence

(c) 2D Persistence

(c) 3D Persistence

(e) Betti Numbers of Sublevel Sets

Fig. 5 Persistence diagrams of a beating heart

References

1. Bajaj, C.L., Pascucci, V., Schikore, D.: The contour spectrum. In: Proceedings of IEEE Visualization, pp. 167–174 (1997)
2. Bendich, P., Edelsbrunner, H., Kerber, M.: Computing robustness and persistence for images. In: Proceedings of IEEE Visualization, vol. 16, pp. 1251–1260 (2010)
3. Biasotti, S., Cerri, A., Frosini, P., Giorgi, D., Landi, C.: Multidimensional size functions for shape comparison. J. Math. Imaging Vis. **32**(2), 161–179 (2008)
4. Carr, H., Snoeyink, J., van de Panne, M.: Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. Comput. Geom. **43**(1), 42–58 (2010)
5. Chen, C., Kerber, M.: An output-sensitive algorithm for persistent homology. In: Proceedings of the 27th annual symposium on Computational geometry 207–215 (2011)
6. Chen, C., Kerber, M.: Persistent homology computation with a twist. In: 27th European Workshop on Computational Geometry (EuroCG 2011) (2011)
7. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of persistence diagrams. Discrete Comput. Geom. **37**(1), 103–120 (2007)
8. Computer Assisted Proofs in Dynamics: CAPD Homology Library, http://capd.ii.uj.edu.pl.
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT, MA (2009)
10. Edelsbrunner, H., Harer, J.: Computational topology, an introduction. American Mathematical Society, RI (2010)
11. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. Discrete Comput. Geom. **28**(4), 511–533 (2002)
12. Freedman D., Chen C.: Algebraic topology for computer vision, Computer Vision. Nova Science (to appear)
13. Freudenthal, H.: Simplizialzerlegungen von beschränkter Flachheit. Ann. Math. **43**(3), 580–582 (1942)
14. Gyulassy, A., Natarajan, V., Pascucci, V., Hamann, B.: Efficient computation of morse-smale complexes for three-dimensional scalar functions. IEEE Trans. Vis. Comput. Graph. **13**(6), 1440–1447 (2007)
15. Kaczynski, T., Mischaikow, K., Mrozek, M.: Computational homology, vol. 157 of Applied Mathematical Sciences. Springer, Berlin (2004)
16. Kedenburg, G.: Persistent Cubical Homology. Master's thesis, University of Hamburg (2010)
17. Kershner, R.: The number of circles covering a set. Am. J. Math. **61**(3), 665–671 (1939)
18. Milosavljevic, N., Morozov, D., Skraba, P.: Zigzag persistent homology in matrix multiplication time. In: Proceedings of the 27th annual symposium on Computational geometry (2011)
19. Morozov, D.: Dionysus: A C++ library for computing persistent homology. http://www.mrzv.org/software/dionysus/
20. Morozov, D.: Persistence algorithm takes cubic time in worst case. BioGeometry News. Dept. Comput. Sci., Duke Univ., Durham, NC (2005)
21. Mrozek, M., Wanner, T.: Coreduction homology algorithm for inclusions and persistent homology. Comput. Math. Appl. 2812–2833 (2010)

22. Mrozek, M., Zelawski, M., Gryglewski, A., Han, S., Krajniak, A.: Homological methods for extraction and analysis of linear features in multidimensional images. Pattern Recogn, **45**(1), 285–298 (2012). doiL 10.1016/j.patcog.2011.04.020
23. Munkres, J.R.: Elements of Algebraic Topology. Addison-Wesley, CA (1984)
24. Pascucci, V., Scorzelli, G., Bremer, P.-T., Mascarenhas, A.: Robust on-line computation of reeb graphs: simplicity and speed. ACM Trans. Graph. **26**(58), 1–8 (2007)
25. Strömbom, D.: Persistent homology in the cubical setting: theory, implementations and applications. Master's thesis, Luleå University of Technology (2007)
26. Zong, C.: What is known about unit cubes. Bull. Am. Math. Soc. **42**(2005), 181–211 (2005)

Part III
Visualization of dynamical systems, vector and tensor fields

Visualizing Invariant Manifolds in Area-Preserving Maps

Xavier Tricoche, Christoph Garth, Allen Sanderson, and Ken Joy

1 Introduction

The Hamiltonian description of dynamical systems, and the formalism arising from it, applies to a large number of natural phenomena from areas as diverse as quantum mechanics, orbital mechanics, fluid dynamics, molecular dynamics, and ecology. At the heart of the Hamiltonian formalism is the principle of *stationary action*, stating that a single scalar function – the *Hamiltonian* – entirely dictates the evolution of a system. In this context, so-called *maps* that describe successive discrete states of an evolving dynamical system and that are used to analyze its structure, can be show to have the property of area preservation, and are extremely rich in structure. From the point of view of scientific visualization, area-preserving maps are simultaneously fascinating and difficult to study as the exhibit fractal topological structure and regions of chaotic behavior.

Due to the widespread prevalence of Hamiltonian systems in applications, an interest in reliable analysis and visualization of area-preserving maps, needed to obtain insight into the fundamental nature of the described system, is found in many scientific disciplines. However, the intrinsic complexity of such maps makes their analysis challenging. So-called *puncture plots* – direct depictions of map iterates in Poincaré sections – offer a straightforward means to obtain a rough picture of the topological features, but are very limited in their ability to offer a reliable picture

X. Tricoche (✉)
Purdue University, West Lafayette, IN 47907, USA
e-mail: xmt@purdue.edu

C. Garth · K. Joy
University of California Davis, Davis, CA 95616, USA
e-mail: cgarth@ucdavis.edu; joy@ucdavis.edu

A. Sanderson
University of Utah, Salt Lake City, UT 84112, USA
e-mail: allen@sci.utah.edu

of the main structures. Moreover, puncture plots are not a reliable tool to discover a priori unknown structures, and inferring the topology of a system from such plots is typically challenging. Despite the introduction of several techniques to address these shortcomings, the effective analysis of maps remains a difficult task.

We present in this paper a new method for the effective visualization of the main topological structures present in area preserving maps. Specifically, we apply in this setting the concept of finite-time Lyapunov exponent to reveal the invariant manifolds of the topology that form the key geometric structure of the map. Our method offers a clear picture of the island chains, which are the signature of these maps and permits to monitor their qualitative evolution over the course of a time-dependent phenomenon.

We apply our approach to a practical scenario and show its application to a numerical simulation of magnetic confinement in a *Tokamak* fusion reactor. It is important to note that we restrict our considerations to *near-integrable systems*. In other words our method is not meant to process fully stochastic systems. The basic premise of our method is that the most significant features of the map can be captured through relatively simple geometry, an assumption that is no longer valid if the system is dominated by chaos. In fact, from a practical standpoint (e.g., in *magneto-hydrodynamics* (MHD) simulations), the ability to characterize topological transformations in the early stages of the simulation is key since it provides a crucial insight into the long term evolution of the system (loss of stability, loss of confinement, etc.). It is worthwhile to point out however that even in seemingly fully ergodic regions, structures can be identified that control the apparently random behavior of the system.

The contents of this paper are organized as follows. Basic definitions and theoretical results relevant to the presentation of our method are first introduced in Sect. 2. We then briefly review previous work on map visualization and analysis in Sect. 3 before describing the algorithmic details of our new method in Sect. 4. Results are presented in Sect. 5. Specifically, we consider the important special cases of the *standard map* (Sect. 5.1) before commenting on our experience with Tokamak simulation data in Sect. 5.2. Finally conclusion and future research directions are discussed in Sect. 6.

2   Theory and Numerical Considerations

We summarize hereafter basic results pertaining to Hamiltonian systems that are needed in the exposition of our work. An excellent introduction to the corresponding theory can be found in classical references [14, 19].

2.1   Hamiltonian Systems and Area-Preserving Maps

A wide variety of physical systems can be described mathematically through Hamilton's equations,

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}, \qquad \frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}. \tag{1}$$

The *state* of the system is entirely described by the point $\mathbf{z}(t) = (p_1, \ldots, p_N, q_1, \ldots, q_N)$, in the $2N$-dimensional *phase space*. The p_i are the *momenta*, q_i are the *positions*, and the scalar function $H(\mathbf{p}, \mathbf{q}, t)$ is called the *Hamiltonian* and typically describes the total energy of the system.

These equations lead to an ordinary different equation (ODE):

$$\frac{d\mathbf{z}}{dt} = \mathbf{f}(z), \;\; \mathbf{z}(0) = \mathbf{z_0}, \tag{2}$$

whereby $\mathbf{z_0}$ constitute the initial condition. The solution $\mathbf{z}(t, \mathbf{z}_0)$ can be associated with a *flow map* $\{\boldsymbol{\phi}_t\}_{t \in I\!R}$, which describes the transport induced in phase space by the dynamical system:

$$\boldsymbol{\phi}_t(\mathbf{z}_0) := \mathbf{z}(t, \mathbf{z}_0), \;\; \boldsymbol{\phi}_0(\mathbf{z}_0) \equiv \mathbf{z_0}. \tag{3}$$

In the following we consider Hamiltonian systems with two degrees of freedom (of the form $\mathbf{z} = (p_1, p_2, q_1, q_2)$) that are invariant under the motion. In this case, one of the variables can be expressed as a function of the other three such as $p_2 = p_2(p_1, q_1, q_2, H = E)$. This effectively transposes the problem to a three-dimensional coordinate system where the motion is confined to a so-called *invariant torus* [14]. We then construct the *Poincaré map* by first selecting a transverse Poincaré section $\Pi = \{q_2 = 0\}$ on which points are described by their coordinates $\mathbf{x}(x, y) := (p_1, q_1)$. By following the trajectory from this point, we define the Poincaré map, $\mathscr{P}$, via $\tilde{\mathbf{x}} = \mathscr{P}(\mathbf{x})$, where $\tilde{\mathbf{x}}$ is the first return point of the trajectory emanating from $\mathbf{x}$ to the plane Π, see Fig. 1.

A fundamental property of Hamiltonian systems compared to other dynamical systems is that the volume of a transported region in the phase space is preserved under the flow map. Hence, the Poincaré map itself is *area-preserving* and the vector field describing the transport induced by the map is *divergence-free* [21]. This observation has profound numerical implications since the interpolation schemes that are typically used in visualization literature will generally not preserve that property of the vector field. We return to this topic in Sect. 2.3.

2.2 *Qualitative Behavior of Hamiltonian Systems*

In the simplest case of Hamiltonian motion (referred to as *integrable case*), the motion is completely ordered. Specifically, the orbits $\mathbf{z}(., \mathbf{z}_0)$ are either closed (and therefore correspond to a periodic motion) or they are confined to tori that are themselves invariant under the flow map. In the Poincaré section these tori form
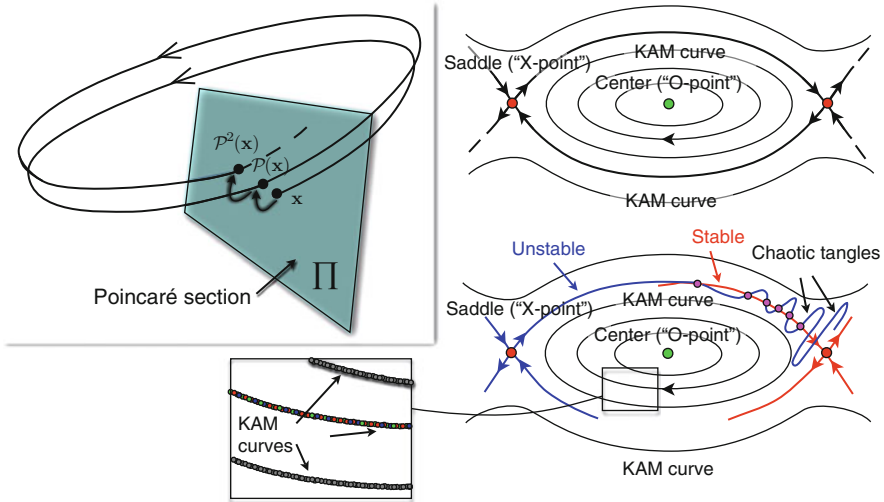
Fig. 1 *Top left:* Two iterations of a Poincaré map. *Right:* Islands of resonance. *Top:* Integrable case. Separatrices connect saddle points in Poincaré map, forming the boundary of an island containing a center point. *Bottom:* Chaotic case. The connections are replaced by the intersection of stable and unstable manifolds forming the tangles that characterize chaos. Quasi-periodic orbits exist both inside and outside of the island and densely populate KAM manifolds (*bottom left*)

nested closed curves. At the other extreme of the qualitative spectrum, Hamiltonian systems exhibit *ergodic* behavior and the motion is random.

The term *chaotic* systems in contrast refers to systems that are neither fully ordered nor fully chaotic. Unsurprisingly, these systems are typical in practice. A defining objective of their analysis is to understand how the canonical structures of the integrable case progressively break under increasing chaos to give rise to the complex patterns observed in the chaotic regime. Among them, so-called islands appear in the phase portrait and irregular, seemingly random trajectories emerge that wander across circumscribed regions of phase space known as *ergodic sea*. Successive iterations of these orbits eventually bring them arbitrarily close to any position within those regions.

2.2.1 Periodic Orbits

By definition, fixed points of the Poincaré map correspond to periodic orbits. A periodic orbit of period p returns to its initial position after p iterations of the map: $\mathscr{P}^p(\mathbf{x_0}) := \mathscr{P}(\mathscr{P}^{p-1}(\mathbf{x_0})) = \mathbf{x_0}$. Here p is uniquely defined as the smallest integer that satisfies this relation since the property trivially holds for any $p' = kp$. Similar to what is known from the analysis of critical points in vector fields, the type of a non-degenerate fixed point can be determined by a local linear analysis of the Poincaré map in its vicinity. This analysis around a

position $\mathbf{x_0}$ is based on the properties of the Jacobian $\mathbf{J}_p := \nabla_{\mathbf{x}} \mathscr{P}^p$, whereby the eigenvalues λ_i, $i = \{1, 2\}$ of $\mathbf{J}_p$ determine the nature of the fixed point. If they are complex conjugates the Poincaré map is characterized by an elliptical motion near the fixed point. This "O-point" configuration is usually referred to as *center* in the visualization literature [11]. A local *island* confining this region of regular motion is present (Fig. 1, top right). If the eigenvalues of the Jacobian are real and of opposite sign, $\mathbf{x_0}$ is associated with a *saddle* pattern (or "X-point") and the eigenvector of $\mathbf{J}_p$ associated with the negative (resp. positive) eigenvalue are tangent to the *stable* (resp. *unstable*) *manifolds* of $\mathbf{x_0}$ that constitute the boundary of the islands and are the *separatrices* of the topology. Saddle points and their invariant manifolds are intimately associated with the chaos displayed by Hamiltonian systems. Note that the successive intersections of stable and unstable manifolds form so-called *chaotic tangles* that shape the dynamics in the chaotic sea, see Fig. 1, bottom right.

2.2.2 Quasi-Periodic Orbits and KAM Theory

Beside fixed points, islands, and ergodic seas, the Poincaré map exhibits curves that are densely covered by quasi-periodic orbits. It follows that the period of these orbits is *irrational* and the fundamental KAM theorem [1, 12, 22] states that those *KAM surfaces* that are "sufficiently irrational" will survive the onset of chaos through nonlinear perturbations. KAM surfaces form perfect transport barriers in the phase portrait, which explains their fundamental importance in physical problems related to confinement. As chaos increases these surfaces are progressively destroyed and replaced by so-called *Cantor sets*, which offer only partial barrier to transport.

2.3 Numerical Aspects

The analysis of an area preserving map depends heavily on an accurate and efficient integration of the flow map $\phi_{t\,t\in I\!R}$. This computation yields the successive iterates of the Poincaré map $\mathscr{P}^i$, $i \in \{1, .., N\}$. An exception to this rule are discrete analytical maps where an explicit formula $\mathbf{f}$ describes the relationship $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x_n})$. We consider one such map in Sect. 5.1. In general, however, the computation of the Poincaré map is made challenging by the need to maintain long term accuracy in the numerical integration of an ODE. In the context of Hamiltonian systems in particular, the property of area-preservation is essentially impossible to guarantee through conventional integration schemes such as Runge-Kutta methods [18]. So-called geometric (or symplectic) integrators do explicitly enforce the invariance of these properties along the integration [7]. However, their application requires a specific formulation of the dynamics (e.g., an explicit expression for the Hamiltonian of the problem), which is rarely available in numerical simulations. When processing such datasets, the continuous reconstruction of the field through piecewise polynomial functions is not exactly conservative. In this case, the area-preserving property is

a theoretical reference for the behavior of the studied phenomenon rather than a numerical reality. For this work, we applied the divergence cleaning approach based on *Hodge projection* advocated by Peikert and Sadlo [23]. However we found the overhead caused by the additional piecewise linear divergence-free interpolation they proposed to outweigh the accuracy benefit. Practically, we used in this work the classic Runge-Kutta triple Dormand-Prince DP6(5) method [25] whose dense output provides an excellent balance of accuracy and speed. However, we found it necessary to require very low relative tolerance of the integration scheme ($\varepsilon = 10^{-8}$) in order to achieve satisfactory results.

3 Previous Work

While discrete dynamical systems and area-preserving maps are not commonly addressed in visualization publications, a number of previous contributions provide the foundations of our method. We briefly summarize them next.

The topological approach has been introduced in visualization by Helman and Hesselink who first showed that the topological skeleton offers a schematic and insightful picture of a (continuous) flow. Numerous contributions have since been made to that general methodology and it remains an active research area [9, 10, 31]. Closer to the topic of this paper, Löffelmann et al. proposed several methods for the intuitive visualization of discrete dynamical systems defined analytically [15–17]. In particular, these authors devised representations that aim at revealing the continuous structures underlying the map. Most recently, Peikert and Sadlo applied a Poincaré map approach to the visualization of vortex rings in a flow recirculation bubble [23, 24]. While the resulting map is not strictly area-preserving, they showed that the topological structures that arise in this context are fundamentally similar. Therefore, they proposed an image-based method to visualize the separatrices that originate from the saddle points located at each extremity of the recirculation bubble and used them to reveal the convoluted patterns formed by their successive intersections along with the associated island chains and stochastic behavior. They also described an iterative scheme to compute the *O-points* located in each of the main islands by successive approximation of the location rotation pattern of the map.

Hamiltonian maps have also been studied in the artificial intelligence and data mining. In his Ph.D. thesis, Yip developed a computer system (called *KAM*) [33] that combines geometric and graph theory criteria commonly used in artificial intelligence to automatic identify the three main types of orbits present in a map: closed loop, island chain, and separatrix. Following a similar approach, Bagherjeiran and Kamath applied a data mining approach to identify patterns in Poincaré plots [2]. Practically, a minimum spanning tree followed by a clustering step is used to infer the 1-dimensional structure of a series of puncture points. Both of these approaches are best suited for the detection of rather large structures of the map. Additionally they do not provide the explicit boundaries of the structures but rather aim at detecting the main features.

In the specific context of fusion reactor simulations, which we consider in Sect. 5.2, Sanderson et al. recently presented a method that automatically constructs a geometric approximation of 1-manifolds in Poincaré maps by resolving the connectivity of the discrete plot [30]. Note that this technique does not explicitly identify invariant manifolds of the topology but instead focuses on the irrational (KAM) surfaces sampled by the seeding. The available curve geometry can then be leveraged to approximate the location of the O-points [24, 29]. The visualization is constructed by probing the map at a discrete set of locations and takes advantage of inherent symmetries in the Tokamak.

Finally, researchers in physics and applied mathematics have considered maps from an algorithmic perspective. England et al. recently proposed a method to construct stable and unstable manifolds in Poincaré maps from saddle points by successively extending the piece of manifold that has already been computed [3]. Again, a similar approach was used in [24]. Levnajić and Mezić considered the application of the ergodic partition theory to the visualization of the standard map [13], whereby their method yields an image in which a piecewise constant color plot reveals coherent regions of the phase space. Most germane to the ideas presented in this paper is the work carried out simultaneously by Grasso et al. [6] who applied FTLE and LCS (Sect. 4) to identify transport barriers in magnetic fields used in plasma confinement. We concentrate hereafter on the visualization implications of this approach and study in more detail the relationship of LCS with the underlying fractal topology.

4   Proposed Approach

We saw in Sect. 2 that the structure of Poincaré maps in Hamiltonian systems can be readily described in topological terms. Hence, it would seem natural to resort to the algorithmic framework of topological methods to create insightful representations of these systems. Unfortunately, this approach proves fantastically difficult in the context of numerical data. Indeed, finding the fixed points of the map is a challenging task that requires a very dense and expensive sampling of the phase portrait. Once fixed points have been identified the next hurdle consists in characterizing the linear type of the fixed point which requires to compute the associated Jacobian. Estimating this derivative properly is also challenging given the chaotic behavior that is unavoidably present in the vicinity of hyperbolic (saddle) points. Finally, the construction of the invariant manifolds associated with the saddles is problematic from a numerical standpoint since topologically they correspond to homoclinic or heteroclinic connections that are known to be generically unstable.

We therefore propose to capture these manifolds in a numerically robust way through the computation of the finite-time Lyapunov exponent (FTLE) in these maps. By yielding an image in which separatrices of the topology are revealed as ridges of the FTLE field this approach provides a powerful means to identify salient geometric structure in a period-agnostic way.

4.1 Finite-Time Lyapunov Exponent

Haller in his seminal work [8] popularized the concept of finite-time Lyapunov exponent to the engineering and visualization community by defining *Lagrangian coherent structures* (LCS) as ridges of the FTLE. Following his approach, unstable (resp. stable) invariant manifolds are characterized as height ridges of FTLE in forward (resp. backward) direction.

Practically, one considers at instant t_0 the flow map $\mathbf{x}_T$, whereby $T = t_0 + \tau$ and τ is the time interval considered for the flow transport. The variations of this flow map around a given position $\mathbf{x_0}$ are determined by its Jacobian $J_\mathbf{x}(t, t_0, \mathbf{x_0}) := \nabla_{\mathbf{x_0}} \mathbf{x}(t, t_0, \mathbf{x_0})$ at $\mathbf{x_0}$ and the maximal rate of dispersion of particles located around $\mathbf{x_0}$ at t_0 is given by following expression ($\lambda_{\max}$ designates the largest eigenvalue):

$$\sigma_\tau(t_0, \mathbf{x_0}) := \sqrt{\lambda_{\max}(J_\mathbf{x}(t, t_0, \mathbf{x_0})^T J_\mathbf{x}(t, t_0, \mathbf{x_0}))}.$$

The average exponential separation rate $\lambda(t, t_0, \mathbf{x_0})$, for positive or negative τ, is then called finite-time Lyapunov exponent and defined [8] as

$$\lambda(t, t_0, \mathbf{x_0}) = \frac{1}{|\tau|} \log \, \sigma_\tau(t_0, \mathbf{x_0}).$$

Ridges of λ for forward (resp. backward) advection correspond to unstable (resp. stable) manifolds that strongly repel (resp. attract) nearby particles. Note that this technique has attracted a significant interest in the visualization literature in recent years [4, 5, 26–28, 32].

It is important to note that, in the presence of a canonical reference frame, these ridges typically capture the separatrices of the topology of steady vector fields [8], owing to the hyperbolicity of the trajectories in the vicinity of the separatrices. Hence the FTLE approach offers a promising alternative to the standard topological method to study the salient structures exhibited by area-preserving maps. In addition, the FTLE method is more robust to noise since it defines structures as the ridge surfaces of a continuously varying coherence measure. Therefore it automatically quantifies the *fuzziness* of the extracted manifolds in the context of chaotic motion. Algorithmic and numerical aspects of this strategy are discussed next.

4.2 Computing FTLE in Maps

A first problem when trying to extend the definition of FTLE in maps is the discrete nature of the dynamics that they describe. Indeed, the notion of finite-time must be expressed in a setup where time is, at best, a discrete notion. Note that in the context of the magnetic field considered in Sect. 5.2, the (integration) time is also available as a continuous dimension. However it makes more sense from a physical

standpoint and for the sake of the corresponding analysis to consider the discrete time associated at each particle with complete revolutions around the system.

The problem posed by discrete time can be solved by defining FTLE at a given location $\mathbf{x_0}$ as:

$$\lambda(k, 0, \mathbf{x_0}) = \frac{1}{|k|} \log \sigma_k(0, \mathbf{x_0}), \text{ with } k \in I\!N.$$

In other word, one replaces time by a number of iterations of the map in the previous definition of FTLE. However, when the map is derived from a continuous system, this definition amounts to a reparameterization of the orbits of the system such that all trajectories complete a full revolution in constant and uniform time. This transformation is a standard procedure in the study of Hamiltonian dynamics where it leads to so-called systems with one and a half degrees of freedom [21].

Another difficulty consists in identifying a proper time scale (τ or k) for the characterization of the structures. While this problem is inherently associated with Haller's FTLE definition it is particularly salient in the case of maps because the fractal complexity of the topology leads to manifolds whose associated time scales vary dramatically across the phase portrait. To tackle this problem we experimented with several approaches. The first one simply consists in integrating the map for a large enough number of iterations to "sharpen" even the small (visible) structures. A clear downside of this approach is that the structures associated with a shorter "time" scale (typically the larger ones) become extremely noisy in the resulting images as aliasing becomes pronounced in their vicinity. An alternative solution consists in computing a FTLE image for each step of a large number of iterations and applying some image processing technique on the resulting stack of image to obtain the best feature characterization. This approach shares some conceptual similarities with the notion of scale space whereby here the the number of map iterations forms a discrete scale axis. A simple such operation consists in identifying at each pixel of the created map the value k such that some image quality metric is maximized. Examples include the value of FTLE or the corresponding ridge strength. Unfortunately this approach provides no guarantee to yield a spatially coherent (e.g., smooth) scale picture since the decision is made on a per-pixel basis. The other shortcoming of this solution is that it requires a large number of intermediate images to be stored as the total number of iterations of the map is computed in order to offer a fine enough sampling of the scale axis. Overall this approach offered some disappointing results although it seems worthwhile to investigate further in future work. We compare the results of these various approaches in Sect. 5.

5   Results

We present in this section results obtained for an analytical and a numerical dataset, respectively. In each case, the computed FTLE values are displayed using a color map that was previously described in [5]. Its basic idea is to favor a clear distinction

between stable and unstable hyperbolic behavior through a distinction between blue and red colors while encoding the relative strength of this behavior through the brightness of the color (stronger values yield darker colors). Aliasing issues caused by the fractal nature of the topology are addressed through smooth downsampling of high resolution maps. The computation of both flow map and FTLE is carried out in on a 32-core Intel "Nehalem" machine, leveraging the embarrassingly parallel nature of the problem. Note that we also implemented the method on the GPU but found the limited accuracy in this case to be unsuitable for our purpose.

5.1 Standard Map

The standard map (also known as *Chirikov-Taylor map*) is an area-preserving 2D map of the 2π square onto itself defined as follows:

$$p_{n+1} = p_n + K \sin(\theta_n) \tag{4}$$

$$\theta_{n+1} = \theta_n + p_{n+1}, \tag{5}$$

whereby p_n and θ_n are taken modulo 2π. K is a parameter that controls the nonlinearity of the map. The standard map describes the dynamics of several mechanical systems and has attracted the attention of theoretical and computational research alike since it is a simple yet powerful tool to study Hamiltonian chaos. This map allows us to test our proposed method across a range of configurations.

Figure 2, left, shows the invariant manifolds of the standard map with $K = 0.7$ as computed for 50 iterations of the map. It can be seen that the FTLE-based visualization clearly reveals the individual island chains of the map. A range of spatial scales are present in this representation that confirm the fractal complexity of the topology. Another compelling property of this representation is its ability to convey the chaos that surround the saddle points of the map. In particular, the saddle of period 1, visible at the top and bottom of the domain (by periodicity) exhibit a typical picture of chaotic tangle. Similarly, this feature is noticeable at both saddles of the period-2 island chain that runs through the middle of the map. These images in fact echo the observations made by Mathur et al. [20] in which an intricate picture of LCS manifolds where shown to underly the apparently chaotic behavior of a turbulent flow.

The application of this approach to an analytical map offers the opportunity to investigate the fractal nature of the topology at arbitrary resolution, only limited by the machine precision. A close-up view in the vicinity of the saddle point visible at the top and bottom of the previous image is proposed in Fig. 2, right, which reveals how subtle structures are properly captured by our method despite their challenging complexity. In particular, the chaotic tangle that is a hallmark of chaos in such systems is prominently present in this image. A comparison with a standard puncture plot offers a contrasting view of these structures.
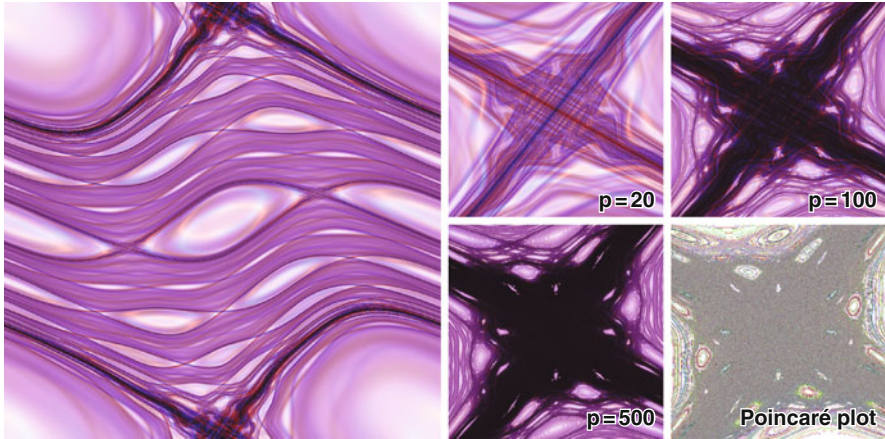
Fig. 2 *Left*: Topology of the standard map for K = 0.7: the map begins to exhibit chaotic regions. *Right*: Invariant manifolds in the vicinity of the 1-saddle as extracted by our method using varying maximal periods. The chaos visible in the overall image is revealed as the product of chaotic tangles. The increasing maximal period count reveals an increasing number of separatrices that form the chaotic tangle

Poincaré plot   FTLE, p = 5000
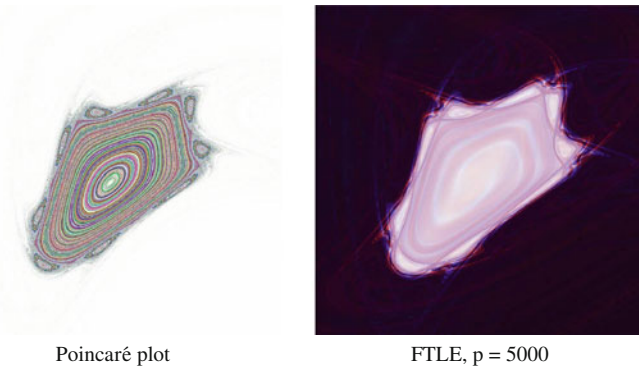
Fig. 3 500× magnified region of the standard map (K = 1.1). The visible island is embedded in the chaotic region, which appears as the region of maximum separation (*right image*)

By increasing the magnification, additional structures become visible, such as those shown in Fig. 3. Note that to achieve a dense enough coverage of such a small region of the map, the puncture plot that is shown here for reference requires an extremely high number of iterations, which goes at the expense of the numerical accuracy of the resulting computation.

5.2 MHD Simulation of Plasma Confinement in a Tokamak Fusion Reactor

Magnetic fusion reactors, such as the International Thermonuclear Experimental Reactor (ITER), a Tokomak reactor scheduled for completion in 2018 will be the source for future low cost power. In their basic operation, magnetic confinement fusion uses the electrical conduction of the burning plasma to contain it within magnetic fields (refer to Fig. 1 in Chap. 23).

A critical characteristic of a typical fusion reactor is the growth of instabilities in the plasma due to the large gradients of density and temperature, the field geometry, and the inherent self-consistent interactions between charged particles and electromagnetic waves. Plasma instabilities occur on very different spatial and temporal scales and can represent highly unique phenomena. One such instability, *magnetic reconnection*, prevents the magnetic field from confining the plasma and leads to its transport. Locating these phenomena can best be done through visualizing the topology of the magnetic field and identifying features within it.

In the normal operation of a tokamak reactor, the magnetic field lines are topologically distinct from each other and form a series of concentric flux surfaces that confine the plasma. Because the magnetic field lines are either periodic, quasi-periodic, or chaotic, the topology can clearly be seen by creating a Poincaré plot. In the presence of instabilities, the magnetic field can become distorted and form magnetic islands. It is the formation of the islands that constitutes magnetic reconnection.

Locating these features; islands, separatrices, and X points, is an important component in understanding plasma transport in magnetic fusion research. However, generating Poincaré plots is computationally expensive and unless seed points for the plot are selected carefully, features within the magnetic field may be missed. As such developing a robust technique that allows for the rapid visualization and facilitates the analysis of topological structures of magnetic field lines in an automatic fashion will aid in the future design and control of Tokamak reactors.

To analyze this time-dependent dataset we first map the computational mesh to its parametric representation in computational space. This amounts to opening up the mesh in both the poloidal and the toroidal direction to yield a 3D mesh in which two directions are periodic. To illustrate some of the aspects discussed previously we first show in Fig. 4 the results obtained in the same dataset for various iterations of the map.

It can be seen that increasing the number of iterations yields a picture in which the finest structures exceed the sampling resolution and lead to significant artifact problems. As mentioned previously, selecting in a spatially varying way the best scale to represent the underlying structures given the limited bandwidth of the image is an open problem for which the solutions that we have investigated so far failed to provide satisfactory results. The close relationship between the FTLE picture and the topology of the Poincaré map is confirmed in Fig. 5.
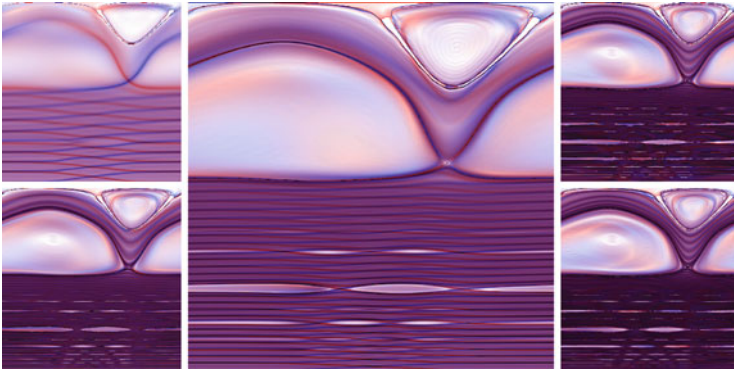
Fig. 4 FTLE mapping of a time step of the Tokamak dataset for various iterations of the map. The central image is obtained for 50 toroidal rotations, while the other ones are obtained for 10, 30, 70 and 100 rotations respectively. The major islands are clearly visible
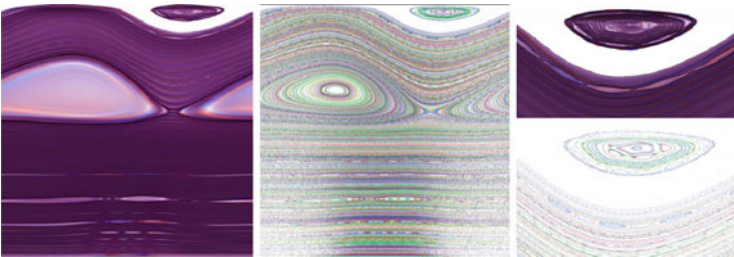

Fig. 5 Comparison between FTLE and Poincaré plot in Tokamak dataset

The images produced by our method lend themselves to an intuitive navigation of the time axis of the simulation. Figure 6 provides such an illustration of the evolution of the topology. In particular it can be seen that a major topological transformation affects the 1-saddle that is visible in the upper part of the domain. This bifurcation known as *basin bifurcation* induces a dramatic reorganization of the transport in the domain.

6 Conclusion

We have presented an algorithmic and computational framework to permit the effective visualization of area-preserving maps associated with Hamiltonian systems. While these maps are of great theoretical interest they are also very important in practice since they offer a geometric interpretation of the structural behavior of complex physical systems. Our method, while conceptually simple and straightforward to implement, significantly improves on previous work by allowing for
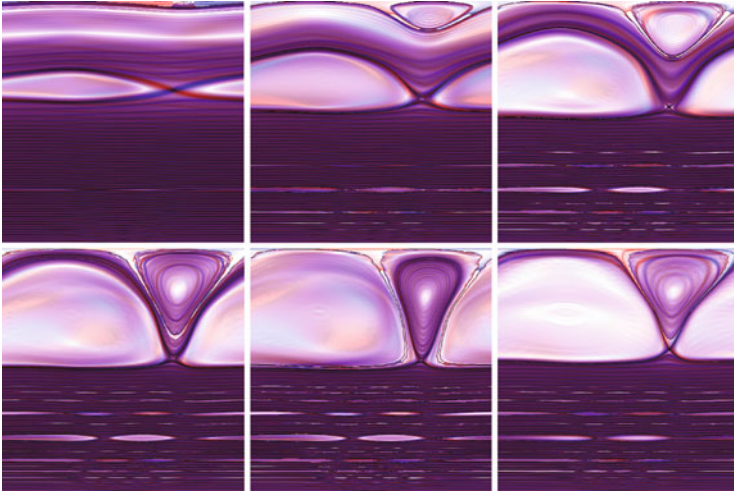
Fig. 6 Temporal evolution of the topological structures present in the map (T = 500, 1,000, 2,000, 3,000, 4,000, 7,000). Each image was obtained after 50 iterations of the map

the identification of very subtle structures that would typically be missed through Poincaré plot investigation of the map. In this context our method successfully addresses one of the primary difficulty posed by this type of structural analysis, namely the numerical challenge associated with an accurate computation of Poincaré maps, which requires the successive integration of an ordinary differential equation. By restricting our computation to a comparatively small number of iterations of the period from any given point (commensurate with the period range relevant to the analysis) we are able to obtain reliable results that are further enhanced by a robust correction strategy motivated by topological considerations.

We have tested our methods on a standard analytical map and on a numerical simulation of magnetic confinement. Our results underscore the potential of our method to effectively support the offline analysis of large simulation datasets for which they can offer a valuable diagnostic tool. In that regard there are many promising avenues for future work. As pointed out in the paper, a proper characterization of the best period to match the spatial scale of the structures would dramatically enhance the results achieved so far. Of course, computational efficiency is a major concern with this method and although it is embarrassingly parallel, adaptive methods, as such recently proposed in the literature, could greatly increase the efficiency of our implementation. Finally, it would be most definitely interesting to combine such a visualization with an explicit extraction of the topology. In that regard, the images obtained suggest that an image processing approach could directly leverage the extracted information while exploiting the theoretical framework of image analysis to do so in a principled way.

References

1. Arnold, V.I.: Proof of A. N. Kolmogorov's thereom on the preservation of quasiperiodic motions under small perturbations of the Hamiltonian. Russ. Math. Surv. **18**(5), 9 (1963)
2. Bagherjeiran, A., Kamath, C.: Graph-based methods for orbit classification. In: Proc. of 6th SIAM International Conference on Data Mining, April 2006
3. England, J., Krauskopf, B., Osinga, H.: Computing one-dimensional global manifolds of poincaré maps by continuation. SIAM J. Appl. Dyn. Syst. **4**(4), 1008–1041 (2005)
4. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Visual. Comput. Graph. **13**(6), 1464–1471 (2007)
5. Garth, C., Wiebel, A., Tricoche, X., Hagen, H., Joy, K.: Lagrangian visualization of flow embedded structures. Comput. Graph. Forum **27**(3), 1007–1014 (2008)
6. Grasso, D., Borgogno, D., Pegoraro, F., Schep,T.: Barriers to field line transport in 3d magnetic configurations. J. Phys. Conference Series 260(1), 012012 (2010). http://stacks.iop.org/1742-6596/260/i=1/a=012012
7. Hairer, E., Lubich, C., Wanner, G.: Geometric Numerical Integration. Springer, Berlin (2006)
8. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional flows. Physica D **149**, 248–277 (2001)
9. Hauser, H., Hagen, H., Theisel, H.: Topology Based Methods in Visualization. Springer, Berlin (2007)
10. Hege, H.-C., Polthier, K., Scheuermann, G.: Topology Based Methods in Visualization II. Springer, Berlin (2009)
11. Helman, J.L., Hesselink, L.: Visualizing vector field topology in fluid flows. IEEE Comput. Graph. Appl. **11**(3), 36–46 (1991)
12. Kolmogorov, A.N.: On the conservation of conditionally periodic motions under small perturbation of the Hamiltonian. Dokl. Akad. Nauk. SSR **98**, 469 (1954)
13. Levnajic, Z., Mezic, I.: Ergodic theory and visualization. I. Mesochronic plots for visualization of ergodic partitions and invariant sets. Chaos **20**(3), 033114 (2010)
14. Lichtenberg, A.J., Lieberman, M.A.: Regular and Chaotic Dynamics, 2nd edn. Springer, New York (1992)
15. Löffelmann, H., Gröller, E.: Enhancing the visualization of characteristic structures in dynamical systems. In: Visualization in Scientific Computing '98, pp. 59–68 (1998)
16. Löffelmann, H., Kucera, T., Gröller, E.: Visualizing poincaré maps together with the underlying flow. In: Hege, H.-C., Polthier, K. (eds.) Mathematical Visualization: Proceedings of the International Workshop on Visualization and Mathematics '97, pp. 315–328. Springer, Berlin (1997)
17. Löffelmann, H., Doleisch, H., Gröller, E.: Visualizing dynamical systems near critical points. In: 14th Spring Conference on Computer Graphics, pp. 175–184 (1998)
18. Marsden, J., West, M.: Discrete mechanics and variational integrators. Acta Numerica **10**, 357–514 (2001). doi:10.1017/S096249290100006X
19. Marsden, J.E., Ratiu, T.: Introduction to mechanics and symmetry. Texts in Applied Mathematics, vol. 17. Springer, Berlin (2003)
20. Mathur, M., Haller, G., Peacock, T., Ruppert-Felsot, J., Swinney, H.: Uncovering the lagrangian skeleton of turbulence. Phys. Rev. Lett. **98**, 144502 (2007)
21. Morrison, P.: Magnetic field lines, hamiltonian dynamics, and nontwist systems. Phys. Plasma **7**(6), 2279–2289 (2000)
22. Moser, J.: On invariant curves of area-preserving mappings of an annulus. Nachr. Akad. Wiss. Göttingen, Math. Phys. Kl. II 1,1 **Kl**(1), 1 (1962)
23. Peikert, R., Sadlo, F.: Visualization methods for vortex rings and vortex breakdown bubbles. In: Museth, A.Y.K., Möller, T. (eds.) Proceedings of the 9th Eurographics/IEEE VGTC Symposium on Visualization (EuroVis'07), pp. 211–218, May 2007

24. Peikert, R., Sadlo, F.: Flow topology beyond skeletons: Visualization of features in recirculating flow. In: Hege, H.-C., Polthier, K., Scheuermann, G. (eds.) Topology-Based Methods in Visualization II, pp. 145–160. Springer, Berlin (2008)
25. Prince, P.J., Dormand, J.R.: High order embedded runge-kutta formulae. J. Comput. Appl. Math. 7(1) 67–75 (1981)
26. Sadlo, F., Peikert, R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. IEEE Trans. Visual. Comput. Graph. **13**(5), 1456–1463 (2007)
27. Sadlo, F., Peikert, R.: Visualizing Lagrangian coherent structures and comparison to vector field topology. In: Hege, H.-C., Polthier, K., Scheuermann, G., Farin, G., Hege, H.-C., Hoffman, D., Johnson, C.R., Polthier, K. (eds.) Topology-Based Methods in Visualization II. Mathematics and Visualization, pp. 15–29. Springer, Berlin (2009)
28. Sadlo, F., Weiskopf, D.: Time-dependent 2-D vector field topology: An approach inspired by Lagrangian coherent structures. Comput. Graph. Forum **29**(1), 88–100 (2010)
29. Sanderson, A., Tricoche, X., Garth, C., Kruger, S., Sovinec, C., Held, E., Breslau, J.: Poster: A geometric approach to visualizing patterns in the poincaré plot of a magnetic field. In: Proc. of IEEE Visualization 06 Conference, 2006
30. Sanderson, A., Cheng, G., Tricoche, X., Pugmire, D., Kruger, S., Breslau, J.: Analysis of recurrent patterns in toroidal magnetic fields. IEEE Trans. Visual. Comput. Graph. **16**(6), 1431–1440 (2010)
31. Scheuermann, G., Tricoche, X.: Topological methods in flow visualization. In: Johnson, C., Hansen, C. (eds.) Visualization Handbook, pp. 341–356. Academic, NY (2004)
32. Soni, B., Thompson, D., Machiraju, R.: Visualizing particle/flow structure interactions in the small bronchial tubes. IEEE Trans. Visual. Comput. Graph. **14**(6), 1412–1427 (2008)
33. Yip, K.M.-K.: KAM: A System for Intelligently Guiding Numerical Experimentation by Computer. MIT, MA (1992)

Understanding Quasi-Periodic Fieldlines and Their Topology in Toroidal Magnetic Fields

Allen Sanderson, Guoning Chen, Xavier Tricoche, and Elaine Cohen

1 Introduction

In magnetic confinement fusion devices such as a tokamak, magnetic fields are used to confine a burning plasma (Fig. 1a). In the study of such devices, physicists need to understand the topology of the *flux surfaces* that form within the magnetic fields. Flux surfaces come in a rational and irrational form and are defined by periodic and quasi-periodic fieldlines, respectively. Our focus is the break up of the rational surfaces into the irrational ones, specifically those that form *magnetic island chains* because they are where the plasma escapes and will damage the wall of the fusion reactor.

To distinguish the magnetic island chains from other flux surfaces, we study their behavior in a Poincaré map computed by intersecting fieldlines with a plane perpendicular to the axis of the torus. A naive geometric test was proposed to identify different magnetic surfaces from the Poincaré section in our previous work [13]. Unfortunately, it does not make use of the periodicity properties of the flux surfaces as we will show later and hence it is computationally expensive and not reliable. In this work we describe a more robust approach that analyzes the distinct periodic behaviors of flux surfaces and island chains, which helps us achieve more efficient characterization of these two structures. More specifically, we estimate the fundamental periods of two functions stemming from the fieldline tracing and the

A. Sanderson (✉) · G. Chen
SCI Institute, University of Utah, Salt Lake City, UT 84112, USA
e-mail: allen@sci.utah.edu; chengu@sci.utah.edu

X. Tricoche
Purdue University, West Lafayette, IN 47907, USA
e-mail: xmt@purdue.edu

E. Cohen
University of Utah, Salt Lake City, UT 84112, USA
e-mail: cohen@cs.utah.edu
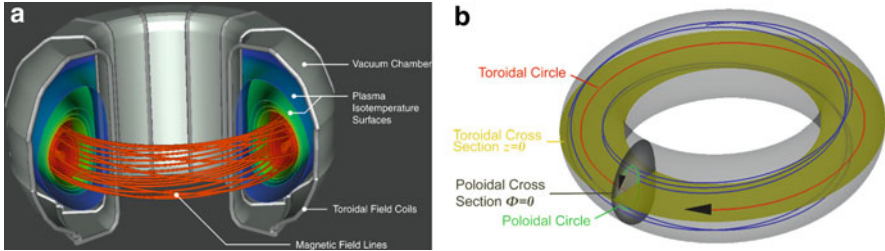
Fig. 1 (**a**) Profile of the DIII-D Tokamak and a single quasi-periodic magnetic fieldline (the *red curves*). (**b**) A magnetic fieldline (*blue*) that intersects the poloidal plane (*gray*) and the toroidal plane (*gold*)

puncture point computation, respectively. These two functions are formed via the *distance measure plot* and the *ridgeline plot*. We show that the periods of these two functions, when coupled are directly related to the topology of a surface. We should note that in our previous work we have described the ridgeline plot but only used it to complement the geometric test. With the addition of the distance measure plot we are now able to couple them together to form a magnetic surface characterization framework which has a more rigorous foundation.

One of the key components in the classification of rational and irrational surfaces, is the *safety factor*. The safety factor is the limit of the ratio of the *winding pair*, i.e., the number of times a fieldline traverses around the major axis of the torus (toroidal winding) for each traversal around the minor axis of the torus (poloidal winding) (Fig. 1b). Although based on the above description the poloidal winding may not be integer (Sect. 4), in the later discussion we consider only integer pairs for the winding pairs. Choosing a good winding pair is of paramount importance because it will determine how we connect the discrete puncture points to get the contiguous representation of the surfaces [13]. In what follows, we will describe in detail how we combine the results of the period estimation of the two functions and other metrics to form a heuristic framework and obtain the desired winding pairs for both geometry construction and topological characterization of a fieldline. We will also explain how the detection of resonance components in the period analysis of an island chain can help us identify this structure in an early stage.

2 Related Work

Magnetic fields are described in terms of vector fields. While a rich body of visualization research has focused on the extraction of features of interest in vector fields [8, 12] starting from [7], the identification of the invariant structures such as periodic orbits from the flow is most relevant to our work.

Within the fusion community researchers have located periodic fieldlines using numerical approaches such as those by [5]. Wischgoll and Scheuermann were the

first in the visualization community to present an algorithm for detecting periodic orbits in planar flows [15]. Their method examines how a fieldline re-enters a cell and re-connects. They have also extended their technique to 3D vector fields [16]. In the meantime, Theisel et al. [14] presented a mesh independent approach to compute periodic orbits. Recently, Chen et al. [2] proposed an efficient algorithm to extract periodic orbits from surface flows using *Morse decompositions*.

We also note the work of Löffelmann et al. [9] who integrated 2D Poincaré plots with the original 3D flow for visualization purposes. Recently, an analysis technique for divergence free flow fields has also been proposed by Peikert and Sadlo [10, 11] who introduced a divergence cleaning scheme to study vortex breakdown flow patterns through their long-term Poincaré plot. Others have used a graph-based approach combined with the machine learning technique to classify the fieldlines [1].

3 Background

In this section, we briefly review some important concepts of Poincaré map and toroidal magnetic fields. More details can be found in [13].

3.1 Poincaré Map

Consider a vector field V on a manifold $\mathcal{M}$ (e.g., a triangulation) with dimension n ($n = 3$ in this work), which can be expressed as an ordinary differential equation $\frac{dx}{dt} = V(x)$. The set of solutions to it gives rise to a *flow* φ on $\mathcal{M}$. Let Γ be a trajectory (integral curve) of a vector field V. Let $\mathcal{S}$ be a cross section of dimension $n - 1$ (e.g., a plane perpendicular to the major axis of the torus) such that φ is everywhere transverse to $\mathcal{S}$. $\mathcal{S}$ is referred to as a Poincaré section. An intersection of Γ with $\mathcal{S}$ is called a *puncture point*, denoted by $l_i \in \mathcal{S} \cap \Gamma$ ($i \in \mathbb{N}$ shows the intersection order). The Poincaré map is defined as a mapping in $\mathcal{S}$ $P : \mathbb{R} \times \mathcal{S} \to \mathcal{S}$ that leads a puncture point l_i to the next position $l_{i+1} \in \mathcal{S} \cap \Gamma$ following Γ.

3.2 Toroidal Magnetic Fields

A toroidal magnetic field is a 3D vector field where the magnetic fieldlines exhibit helical behavior and wind around the major (toroidal) circle and minor (poloidal) circle of the torus (Fig. 1b).

The safety factor of a fieldline, q, is defined as the number of times a fieldline goes around the toroidal circle for each rotation around the poloidal circle, and is computed as:

$$q = \lim_{\#T \to \infty} \frac{\#T}{\#P} \tag{1}$$
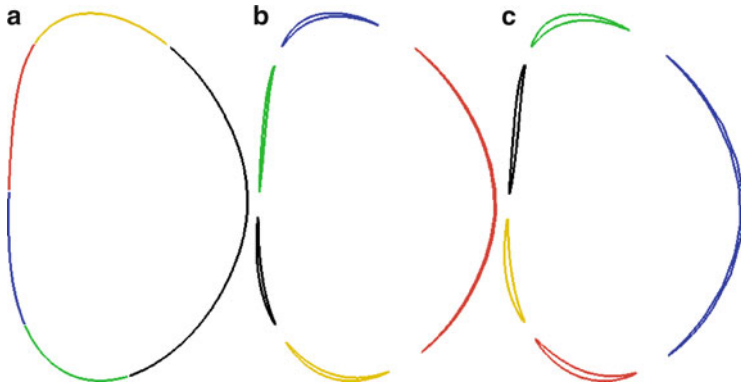
Fig. 2 The evolution of a magnetic surface at different times of the fusion simulation: (**a**) an irrational flux surface at time step 22; (**b**) a 5,2 island chain at time 23; (**c**) the growing 5,2 island chain at time 24

where #T is the toroidal winding count (rotations about the toroidal circle) and #P is the poloidal winding count (rotations about the poloidal circle). We define the two winding counts when expressed as rational numbers as a *winding pair*. Because we cannot integrate to infinity, in practice we estimate the safety factor of a surface based on a finite number of integrations (Sect. 4.1).

By definition, the safety factor q can be either rational or irrational. A rational q implies that the fieldline is periodic (or closed in finite distance). Such a fieldline lies on a *rational surface*. Such surfaces are found in a fusion device. However, they are unstable and sensitive to the magnetic perturbation. Among them, the ones with lower-order q are the first to break down into island chains [3]. Figure 2 provides an example of such a topology change of a magnetic surface due to the magnetic perturbations. An irrational q implies that the fieldline is quasi-periodic. Such a fieldline lies on an *irrational surface* and spreads out over it. This type of flux surfaces is our focus in this work. They have two distinct topological structures shown in a Poincaré section, a single closed curve or multiple closed curves. A single closed curve typically represents a magnetic flux surface when it encloses the center of the magnetic field. Multiple closed curves represent a magnetic island chain which is usually associated with a reduction in magnetic confinement [13].

There are two types of critical points for a magnetic island chain, commonly referred to as X (unstable or saddle) and O (stable or center) points. They correspond to the locations where there is no poloidal magnetic flux [6]. In the case of an O point, it is located at the magnetic center of an island. While for an X point, it is located where two flux surfaces appear to cross and form a *separatrix* around the magnetic islands. In this work, we focus on only the characterization of fieldlines. The detection of critical points can be found in [13].

According to [13], an irrational surface consists of #T winding groups in the Poincaré section. The geometrically neighboring groups need not be neighbors in the puncture point ordering. In order to construct a valid geometry representation

without self-intersections for the surface, a proper winding pair is greatly desired. In the following, we describe how we achieve so.

4 Fieldline Puncture Points and Winding Pairs

In [13] we collected a set of puncture points at the Poincaré section while counting the numbers of the associated toroidal and poloidal windings of the fieldline. We briefly review this collection: Let A_i be a tuple describing the state of each puncture point of a fieldline Γ at the Poincaré section $\mathscr{S}$. Further, let $A_i = (l_i, \#T, \#P)$ where l_i represents the location of A_i in $\mathscr{S}$, $\#T$ is the number of crossing of Γ through $\mathscr{S}$ when reaching l_i, and $\#P$ the number of crossing of Γ through the toroidal cross section (the horizontal brown plane in Fig. 1b) when $d(\Gamma)_z > 0$ (increasing z coordinates) and when reaching l_i (see Fig. 1b for an illustration).

The above gives a good estimation of the poloidal winding when the magnetic axis (the central axis of the magnetic field) is nearly planar. However, as the magnetic field is perturbed, the axis no longer lies in a plane and the above estimation fails. As such, we do a more computationally expensive continuous sampling of the poloidal winding through a rotational transform [3]:

$$\#P \approx \frac{1}{2\pi} \int_o^S ds \frac{d\theta}{ds} \, ds \tag{2}$$

where S is the total length that the fieldline is integrated over and $d\theta$ is the change in poloidal angle for the distance ds traveled along the fieldline. Further $\frac{d\theta}{ds}$ can be defined as:

$$\frac{d\theta}{ds} = \frac{d}{ds}[arctan(\frac{Z}{R})] = \left(\frac{1}{R^2 + Z^2} \right) \left(R \frac{dZ}{ds} - Z \frac{dR}{ds} \right), \tag{3}$$

while in a cylindrical coordinate system. That is, we are summing up the poloidal changes along the fieldline and then dividing by the total toroidal distance traveled.

When reaching l_i in the Poincaré section $\mathscr{S}$ we record $\#T$ like before (as an integer) and $\#P$ from (2) as a rational number (i.e., rounded to an integer). $\#P$ is stored as a rational number because we are interested in two rational values (aka a winding pair) to utilize with the fundamental periods (integers) of the distance measure plot and ridgeline plots that will be described in Sect. 5.

4.1 Safety Factor Approximation

By definition the safety factor is the limit of the ratio of the winding numbers when the fieldline is traced infinitely long (1). As noted, only a limited number of integration steps can be computed before numerical inaccuracies lead the field line to an erroneous path. As such, to approximate the safety factor we simply divide $\#T$ by $\#P$ (as a floating point value) from (2) for the last puncture point in A.

4.2 Ranking Winding Pairs

In [13], we identified a single poloidal-toroidal winding pair for a value of T such that

$$\min_{T \in \mathbb{N}}(d) = \sum_{i=T} \|(P_{i+T} - P_i) - (P_i - P_{i-T})\| \tag{4}$$

is minimized, where P_i is the number of crossings of Γ through the toroidal cross section plane when reaching the puncture point l_i. When d is minimized, the toroidal winding number $\#T = T$ and the poloidal winding number $\#P = P_T$.

The minimization is based on an important observation: for a given toroidal winding number $\#T$, the poloidal winding number should be consistent between every $\#T$ puncture points. For example, if the toroidal winding number is 5 and the poloidal winding number is 2. Then the poloidal winding counts, $\#P$ could be:

0, 1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5, 5, 5, 6.

In this case the difference between every 5th value (the toroidal winding number) is 2 (the poloidal winding number).

While the minimization results in one "best" winding pair there are multiple possible winding pairs, each of which is a *rational approximation* to the irrational safety factor. In our previous work [13] we were only interested in the "best" winding pair while in the present work we are interested in multiple pairs. In Table 1 we show the possible pairs for an irrational surface ranked based on the minimization criteria in (4). For this criterion, a 29,11 surface would be the best candidate.

It is also possible to rank the winding pairs based on other criteria. For example, the 29,11 pair results in a safety factor of 2.63636 which is not the best approximation given the safety factor of 2.6537 as calculated using the rotational sum. As such, selecting the winding pair that is closest to the approximated safety factor is another option. In Table 2, we rank the winding pairs based on this criterion. In this case, the 61,23 surface would be the closest match while the 29,11 surface would now be ranked sixth. However, the winding pair we seek should provide us information for the geometry reconstruction. The winding pairs obtained using the

Table 1 Winding pairs using 125 puncture points, 48 ridgeline points with a base safety factor of 2.6537 ranked based on the best matching pair consistency

Toroidal/poloidal winding pair	Safety factor	Equation (4) (normalized)
29, 11	2.63636	98.9583
37, 14	2.64286	98.8636
8, 3	2.66667	97.4359
45, 17	2.64706	96.25
21, 8	2.625	95.1923
50, 19	2.63158	93.3333
53, 20	2.65	93.0556
13, 5	2.6	91.9643
61, 23	2.65217	90.625

Table 2 Winding pairs using 125 puncture points and 47 ridgeline points with a base safety factor of 2.6537 ranked based on the best rational approximation

Toroidal/poloidal winding pair	Safety factor	Best rational approximation
61, 23	2.65217	0.00152259
45, 17	2.64706	0.00663767
53, 20	2.65	0.0036965
37, 14	2.64286	0.0108394
8, 3	2.66667	0.0129702
29, 11	2.63636	0.0173329
50, 19	2.63158	0.0221176
21, 8	2.625	0.0286965
13, 5	2.6	0.0536965

above two criteria could not be proven to contain such information. Therefore, in the following we turn to the analysis of two functions, the distance measure plot and ridgeline plot.

5 Distance Measure Plot and Ridgeline Plot

In this section, we describe the distance measure and ridgeline plots whose periods are dependent on the toroidal and the poloidal periods, respectively. To obtain their periods and subsequently the classification of the fieldlines, we perform a period analysis. Unlike the safety factor and winding pair analysis which required both the toroidal and poloidal windings, the period analysis of each plot is independent.

5.1 Distance Measure Plot

We introduce a distance measure that is defined as the distance between two puncture points:

$$d_i = \|l_{i+T} - l_i\| \tag{5}$$

where T is the interval between two puncture points.

When the fieldline is periodic (i.e., lies on a rational surface) and has a toroidal period of T, d_i between every T points will be zero. When the fieldline is quasi-periodic (i.e., lies on an irrational surface) d_i will be non zero for all points in A (Fig. 3). But the sum of distances is minimal when T is the toroidal winding period. This is equivalent to finding the period T that minimizes the following:

$$\min_{T \in \mathbb{N}}(d) = \sum_{i=T} \|l_{i+T} - l_i\| \tag{6}$$

Fig. 3 An 8,3 flux surface
with the puncture points, l_i
and the distances, d_i between
each eighth puncture point
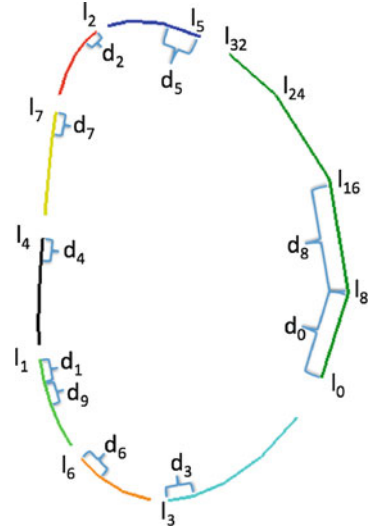
One can interpret (6) as the solution to a minimum spanning tree where the points
are the nodes and the weights of the edges are the distances between the points, d_i.

While perhaps not obvious one can look at each interval T as the basis for a 1D
plot where the sample values are the distances from (5) (thus the term distance
measure plot). Figure 4b, d show two distance measure plots with T equal the
fundamental periods where the sum of the distances are minimized.

5.2 Ridgeline Plot

Previously [13], we noted that for each poloidal winding in the fieldline there is a
local maximum, r with respect to the toroidal cross section (i.e., the $Z = 0$ plane),
which is defined as:

$$\Gamma_z(r) > 0; \frac{\partial \Gamma(r)}{\partial z} = 0. \tag{7}$$

The *ridgeline plot* is defined as the collection of these local maxima.

It is easy to understand the construction of a ridgeline plot when we view the
field lines in cylindrical coordinates (Fig. 5a). The periodic nature of the fieldline is
then apparent (Fig. 5b). The oscillation of the ridgeline can be attributed to an area
preserving deformation of the magnetic surface as the fieldline precesses around it
and its *fundamental period* is the poloidal period of the fieldline.

To extract the fundamental period of a ridgeline plot, which is essentially a
1D function we make use of a Yin Estimator [4] which minimizes the following
difference:
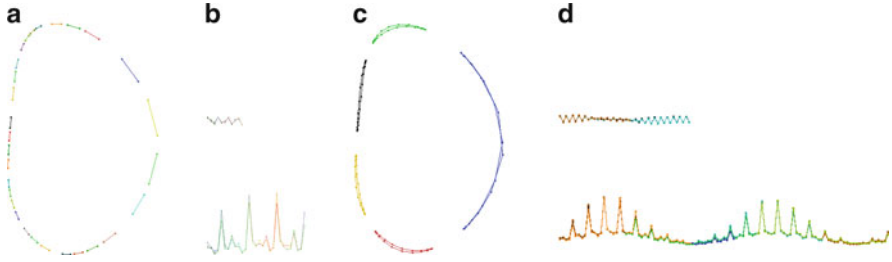
Fig. 4 (**a**) A Poincaré section of a 29,11 flux surface. The number of curved sections, 29 corresponds to the toroidal winding number. (**b**) *Top*, the ridgeline plot with a period of 11. (**b**) *Bottom*, the distance plot with a period of 29. (**c**) A Poincaré section of a 5,2 island chain. The number of islands, 5 corresponds to the toroidal winding number. (**d**) *Top*, the ridgeline plot with a period of 42. (**d**) *Bottom*, the distance plot with a period of 105. Each island contains 21 points in its cross-section
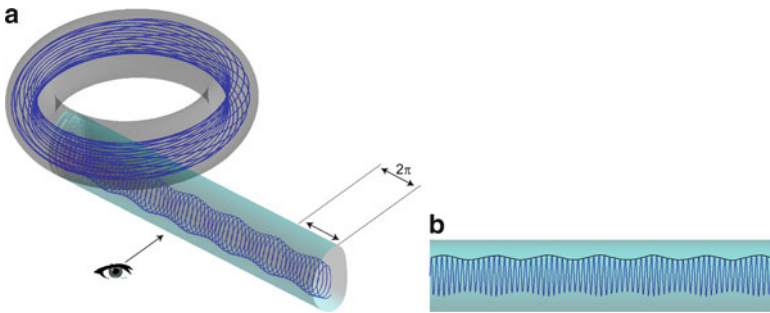
Fig. 5 (**a**) The original toroidal geometry containing a single fieldline for multiple toroidal windings in Cartesian coordinates superimposed with the same geometry in cylindrical coordinates. (**b**) The ridgeline plot of maximal points is shown in *black* and has a period of 10

$$\min_{f \in N}(\sigma) = \sum_{i=0}(r_i - r_{i+f})^2 \tag{8}$$

where r_i is the local maxima from (7), and f is the fundamental period.

5.3 Combining Measures

When analyzing distance measure and ridgeline plots we are able to obtain multiple solutions with different rankings using (6) and (8), respectively. For example, in Tables 3 and 4 we show the candidate periods for the distance (toroidal) and ridgeline (poloidal) plots with the descending ranking respectively. Similar periods are seen in Tables 1 and 2 but with different rankings.

Table 3 Toroidal winding periods via 125 poloidal punctures ranked based on the best period

Toroidal period	Normalized variance
37	0.00255453
58	0.0071927
29	0.00790372
45	0.0177908
50	0.0346803
53	0.0352651
61	0.0433945
63	0.0760976
42	0.0865597

Table 4 Poloidal winding period via 46 ridgeline points ranked based on the best period

Poloidal period	Normalized variance
14	1.07227e–05
22	2.48072e–05
11	2.83332e–05
17	6.16752e–05
20	0.000101897
19	0.000105744
23	0.000129318
16	0.000209808
24	0.000210209

Table 5 Possible winding pairs found in Table 1 using the periods from Tables 3 and 4 ranking based on an Euclidean distance. [1] reduced to 29,11, [2] discarded

Winding pair	Euclidean distance
37,14	0
58, 22[1]	1.41421
29, 11[2]	2.23607
45,17	4.24264
53,20	6.40312
50,19	6.40312
61,23	8.48528
21,8	10.6301
8,3	13.6015

While each of the plots yields independent toroidal and poloidal winding periods we now combine them to form the same winding pairs found in Table 1. Further, we rank each pair based on their individual rankings using an Euclidean distance measure (sum of the squares of their individual rankings) (e.g., Table 5).

This ranking results in winding pair 37,14 being the "best" overall approximation having a Euclidean distance of 3.16228 (in Tables 1, 2, and 5 the 37, 14 winding pair was ranked second, fourth, and first respectively thus $\sqrt{1^2 + 3^2 + 0^2}$). The resulting surface is shown in Fig. 6. We can use this metric regardless of topology albeit not for chaotic fieldlines.

Table 6 Winding pairs ranked based on an Euclidean distance

Winding pair	Euclidean distance
37,14	3.16228
45,17	4.12311
29,11	5.09902
53,20	6.78233
8,3	8.30662
50,19	8.77496
61,23	9.43398
21,8	10.0499
13,5	13.3041

Fig. 6 The best winding pair from Table 6 a 37,14 flux surface with 37 line segments representing the 37 toroidal windings
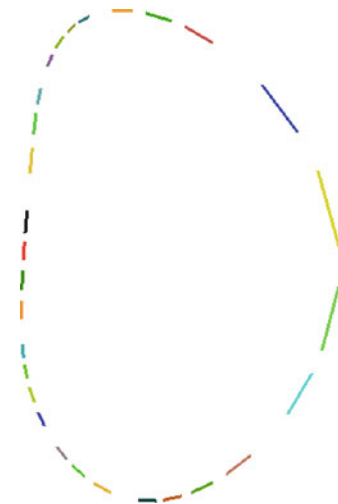
It is worth noting that the 29,11 winding pair also shows up as 58,22 winding pair (see Table 5). The latter can be reduced to a 29,11 pair as the integers 58,22 have a common denominator of 2. Further, the original 29,11 pair is discarded because its Euclidean ranking distance was greater than the 58,22 pair (Table 5).

Winding pairs that share a common denominator are not unexpected. However, as will be discussed in Sect. 6, common denominators are a key to differentiating between the topology of flux surfaces and magnetic islands.

6 Identifying Island Chains Using Period Estimation

Up to this point we have focused solely on identifying a winding pair that approximates an irrational surface. However, we note that in the case of an island chain an interesting phenomenon occurs.

When analyzing a flux surface, the fundamental periods of the distance and ridgeline plots are *equal* to the toroidal and poloidal winding pair values (Fig. 4b). While for an island chain, the fundamental periods of the distance and ridgeline plots are *proportional* to the toroidal and poloidal winding pair values (Fig. 4d). In [13] we showed that the proportionality constant for the ridgeline plot was equal to the number of points in the cross-sectional profile of the island. We have since observed the same for the distance measure plot. This proportionality is due to the periodic nature of the puncture points as well as the periodic nature of the points defining the cross section of the island.

While we have not yet fully investigated, we believe the difference in the proportionality between a flux surface and an island chain is due to the fact that the puncture points in an island chain are topologically separate from each other (i.e., multiple closed curves), while for a flux surface the puncture points will overlap with each other (i.e., one closed curve).

Further, we observed that the number of candidate winding pairs is typically limited as the irrational fieldlines of the island chains continue to reflect the rational fieldlines from which they originated (or broke down) from. For instance, for a 5,2 island chain that utilizes 271 poloidal puncture points and 107 ridgeline points using (4) (Sect. 4.2) results in only one winding pair 5,2 being found.

Constructing the distance measure and ridgeline plots and obtaining the fundamental periods, as shown in Tables 7 and 8, we obtain a "best" winding pair of 105, 42. We can reduce the winding pair to 5,2 by dividing by 21. In the meantime, in Fig. 4c, the cross sectional profile of each island is composed of 21 points. In other words, the greatest common denominator of the two periods in the best winding pair obtained using the combined measures of the distance measure and ridgeline plots is larger than 1 for an island chain (typically larger than 3 in order to form a closed shape). On the other hand, this greatest common denominator is usually 1 for a flux surface (see the 37,14 surface in Table 5). This characteristic provides a simple test for determining whether a surface is a flux surface or an island chain. If the "best" fundamental periods of the distance and ridgeline plots are some integer multiples of the toroidal and poloidal periods found by (4), then not only is the surface an island chain but the common multiplier (e.g., 21 in the previous example) of these multiples is the number of points in the cross sectional profile of each island.

	Toroidal period	Normalized variance
Table 7 Toroidal winding periods for an island chain via 271 poloidal punctures ranked based on the best period	105	0.0001533
	110	0.00197257
	100	0.00428337
	115	0.00836247
	95	0.0143386
	120	0.0164212

Table 8 Poloidal winding period for an island chain via 108 ridgeline points ranked based on the best period

Poloidal period	Normalized variance
42	4.29724e–07
44	4.95931e–06
40	1.14382e–05
46	2.00078e–05
38	3.63677e–05
48	3.85742e–05

An alternative way for determining whether a surface is a flux surface or an island chain is by looking at the common denominator for the lists of toroidal periods and poloidal periods, separately. For instance, the common denominators for the candidate toroidal and poloidal periods in Tables 7 and 8 are 5 and 2 respectively, which is exactly the winding pair found from (4). The common denominator is due to the resonance nature of the fieldline and is an indication of the island topology. For a flux surface, such as for Tables 3 and 4 no such common denominator other than 1 will be found. As will be shown below, this common denominator property gives us an indication of the type of the resonance of the two functions (and literally the fieldline).

7 Results and Discussion

The combined metrics described in Sect. 5.3 have produced accurate results for our tests so far, failing only when encountering chaotic fieldlines. In addition to identifying flux surfaces and magnetic island chains, the metrics are also used to identify rational surfaces whose fieldlines are truly periodic, $A_0(l_0) = A_{\#T}(l_{\#T})$. However, we have found that our technique is not always able to give a definitive result because as one approaches a rational surface the distance between adjacent points goes to zero, which in turn requires an infinite number of points for the analysis. As the future work, we plan to investigate the analysis of rational surfaces using a limited number of points.

We have also examined the periodicity of separatrices near island chains. In Fig. 7, the best rational approximations for the three surfaces are all 2,1. It is purely coincident that all three surfaces had the same approximation as selecting a slightly different starting seed point near separatrix could have resulted in a higher order approximation (i.e., a winding pair with larger integers).

In Sect. 6 two characteristics, i.e., integer multiples and common denominators were discussed which could be used to identify magnetic islands and their unique topology. However, we observed cases where the common denominator did not equal the winding pair found from (4). For instance, for a 3,1 island chain that utilizes 278 poloidal puncture points and 91 ridgeline points has a safety factor of 2.99932 while the only winding pair found using (4) is 3,1. Constructing distance

Fig. 7 A 2,1 magnetic island chain (*red*) surrounded by its separatrices, (*blue* and *black*)
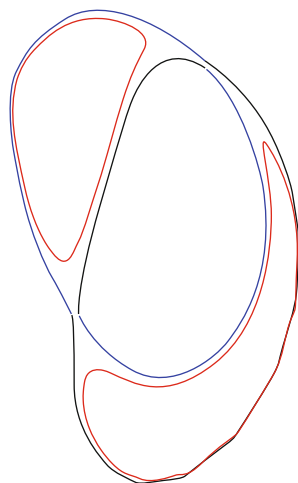
Table 9 Toroidal winding periods for an island chain via 278 poloidal punctures ranked based on the best period

Toroidal period	Normalized variance
108	6.79968e–05
90	0.00092499
126	0.00098725
144	0.00178431
72	0.00195148
54	0.00199423
36	0.00200031
18	0.00201468

measure and ridgeline plots and computing the fundamental periods (Tables 9 and 10) we obtain a "best" winding pair of 108, 36. We can reduce the winding pair to 3, 1 by dividing by 36 which is the number of points in the cross sectional profile.

However, unlike our previous island chain example the common denominators for the toroidal and poloidal periods, 18 and 6 respectively, do not equal 3,1. Though they do reduce down to it. This secondary reduction or more precisely secondary resonance, gives further topological information about the island chain. Specifically, the island chain itself contains islands (aka islands within islands). To reduce the common denominators 18,6 to 3,1 an integer multiple of 6 is required. Which is the number of small islands surrounding each island (Fig. 8) with each island containing 6 points (i.e., 6 islands with 6 points equaling 36, the number of points in the cross sectional profile).

Finally, we note that we do not compare the present results to our previous geometric tests because they were not a general solution, to the point of being ad-hoc in nature. More importantly they required overlapping puncture points in order to obtain a definitive result. Our new technique gives a definitive result as long as

Table 10 Poloidal winding period for an island chain via 91 ridgeline points ranked based on the best period

Poloidal period	Normalized variance
36	Variance 3.70173e–07
30	Variance 4.4525e–06
42	Variance 4.71553e–06
48	Variance 9.01227e–06
18	Variance 1.01154e–05
24	Variance 1.01895e–05
12	Variance 1.03454e–05
6	Variance 1.03557e–05
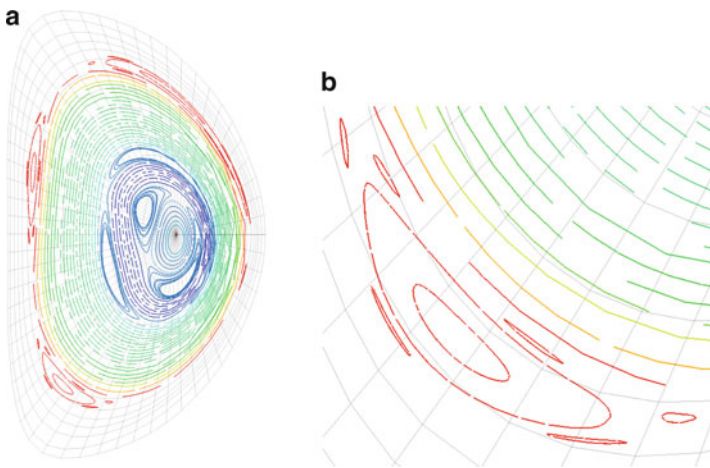
Fig. 8 (**a**) Poincaré plot from a NIMROD simulation of the D3D tokamak. (**b**) A closeup from the lower left showing the island within islands topology. In this case there are six islands surrounding each of the islands that are part of a 3,1 island chain

there is a sufficient number of puncture points to perform the period analysis (i.e., at least twice of the fundamental period).

8 Summary

In this paper, we discuss the period analysis of the quasi-periodic fieldlines in a toroidal magnetic field. We show that the topology of these fieldlines has direct relationship to the fundamental periods of the distance measure plots and ridgeline plots that are obtained through the computation of fieldlines and Poincaré plot, respectively. We have described how the period analysis of these two plots characterize the behavior of the fieldline. The present framework while having its basis in resonance detection relies on a heuristic solution. Therefore, the future work will focus on the further evaluation of the present combined analysis, and

the development of more robust technique for characterizing fieldlines including identifying different topological structures and extracting winding pairs.

References

1. Bagherjeiran, A., Kamath, C.: Graph-based methods for orbit classification. In: SIAM International Conference on Data Mining. SIAM, PA (2005)
2. Chen, G., Mischaikow, K., Laramee, R.S., Pilarczyk, P., Zhang, E.: Vector field editing and periodic orbit extraction using morse decomposition. IEEE Trans. Visual. Comput. Graph. **13**(4), 769–785 (2007)
3. D'haeseleer, W.D., Hitchon, W.G., Callen, J.D., Shohet, J.L.: Flux Coordinates and Magnetic Field Structure, A Guide to a Fundamental Tool of Plasma Theory. Springer, New York (1991)
4. Gerhard, D.: Pitch extraction and fundamental frequency: History and current techniques. Technical report, 2003–06
5. Green, J.M.: Locating three-dimensional roots by a bisection method. J. Comput. Phys. **98**, 194–198 (1992)
6. J. M. Greene.: Vortex nulls and magnetic nulls. In A. T. H.K. Moffatt (ed.) Topological fluid mechanics: proceedings of the IUTAM Symposium, pp. 478-484. Cambridge Univ. Press (1990)
7. Hale, J., Kocak, H.: Dynamics and Bifurcations. Springer, New York (1991)
8. Laramee, R., Hauser, H., Zhao, L., Post, F.H.: Topology based flow visualization: The state of the art. In: Topology-Based Methods in Visualization (Proceedings of Topo-in-Vis 2005), Mathematics and Visualization, pp. 1–19. Springer, Berlin (2007)
9. Löffelmann, H., Kucera, T., Gröller, M.E.: Visualizing poincare maps together with the underlying flow. In: International Workshop on Visualization and Mathematics '97, pp. 315–328. Springer, Berlin (1997)
10. Peikert, R., Sadlo, F.: Visualization methods for vortex rings and vortex breakdown bubbles. In: Museth, A.Y.K., Möller, T. (eds.) Proceedings of the 9th Eurographics/IEEE VGTC Symposium on Visualization (EuroVis'07), pp. 211–218, May 2007
11. Peikert, R., Sadlo, F.: Flow topology beyond skeletons: Visualization of features in recirculating flow. In: Hege, H.-C., Polthier, K., Scheuermann, G. (eds.) Topology-Based Methods in Visualization II, pp. 145–160. Springer, Berlin (2008)
12. Post, F.H., Vrolijk, B., Hauser, H., Laramee, R.S., Doleisch, H.: The state of the art in flow visualization: Feature extraction and tracking. Comput. Graph. Forum **22**(4), 775–792 (2003)
13. Sanderson, A., Chen, G., Tricoche, X., Pugmire, D., S. Kruger, S., Breslau, J.: Analysis of recurrent patterns in toroidal magnetic fields. IEEE Trans. Visual. Comput. Graph. **16**, 1431–1440 (2010)
14. Theisel, H., Weinkauf, T., Seidel, H.-P., Seidel, H.: Grid-independent detection of closed stream lines in 2D vector fields. In: Proceedings of the Conference on Vision, Modeling and Visualization 2004 (VMV 04), pp. 421–428, Nov 2004
15. Wischgoll, T., Scheuermann, G.: Detection and visualization of closed streamlines in planar fields. IEEE Trans. Visual. Comput. Graph. **7**(2), 165–172 (2001)
16. Wischgoll, T., Scheuermann, G.: Locating closed streamlines in 3D vector fields. In: Proceedings of the Joint Eurographics – IEEE TCVG Symposium on Visualization (VisSym 02), pp. 227–280, May 2002

Consistent Approximation of Local Flow Behavior for 2D Vector Fields Using Edge Maps

Shreeraj Jadhav, Harsh Bhatia, Peer-Timo Bremer,
Joshua A. Levine, Luis Gustavo Nonato, and Valerio Pascucci

1 Introduction

Typically, vector fields are stored as a set of sample vectors at discrete locations. Vector values at unsampled points are defined by interpolating some subset of the known sample values. In this work, we consider two-dimensional domains represented as triangular meshes with samples at all vertices, and vector values on the interior of each triangle are computed by piecewise linear interpolation.

Many of the commonly used techniques for studying properties of the vector field require integration techniques that are prone to inconsistent results. Analysis based on such inconsistent results may lead to incorrect conclusions about the data. For example, vector field visualization techniques integrate the paths of massless particles (streamlines) in the flow [25] or advect a texture using line integral convolution (LIC) [2]. Techniques like computation of the topological skeleton of a vector field [9, 10], require integrating separatrices, which are streamlines that asymptotically bound regions where the flow behaves differently [11]. Since these integrations may lead to compound numerical errors, the computed streamlines may intersect, violating some of their fundamental properties such as being pairwise disjoint. Detecting these computational artifacts to allow further analysis to proceed normally remains a significant challenge.

S. Jadhav (✉) · H. Bhatia · J. A. Levine · V. Pascucci
SCI Institute, University of Utah, Salt Lake City, UT 84112, USA
e-mail: jadhav@sci.utah.edu; hbhatia@sci.utah.edu; jlevine@sci.utah.edu; pascucci@sci.utah.edu

P.-T. Bremer
Lawrence Livermore National Lab, 7000 East Ave., Livermore, CA 94550-9234, USA
e-mail: bremer5@llnl.gov

L. G. Nonato
Universidade de São Paulo, Sao Paulo 05508-010, Brazil
e-mail: gnonato@icmc.usp.br

1.1 Contributions

To address this challenge in situations where consistent analysis is required, we propose a new representation of flow through each triangle. This representation, called *edge maps*, approximates the flow by mapping pairs of boundary points that are on the same streamline to each other. Once computed, edge maps replace the numerical integration of streamlines with a lookup, which enforces that computed streamlines never cross. With implementation in mind, we can merge adjacent pairs of mapped points to provide a multi-resolution approximation of the edge map, enabling us to build more compact representations of the flow. At any resolution of merging, the edge maps still guarantee consistent streamlines. At the finest scale of resolution, streamlines remain accurate on triangle boundaries, while at the coarsest scale (the maximum possible amount of merging), the edge map represents one class of possible flow behavior. We enumerate these possibilities and show there are 23 equivalence classes that can exist for triangles in a piecewise linearly interpolated vector field. While the discussion within this work is primarily theoretical, an implementation of edge maps and their applications have been described in [1].

2 Related Work

One step towards producing consistent streamlines is to use higher accuracy integration techniques. Some recent results that improve on the traditional Runge-Kutta based techniques include the local exact method (LEM) of Kipfer et al. [13] that follows the lead of Nielson and Jung [16]. LEM solves an ODE for simplices on unstructured grids representing the position of the particle as a function of time, starting at a given position. While the solution is often more expensive than numeric integration, given an entry point of a particle to a triangle, LEM gives its exact path within the triangle. However, the exit point is calculated numerically as an intersection with the triangle edges which is prone to numeric errors. Hence, consistency of streamlines still cannot be guaranteed. Despite this, it is the most accurate technique available since it does not incur integration error. Although our work focuses on the mathematical properties of edge maps, we mention these results since a system using edge maps could first rely on such a computation for construction. Using LEM we can get a more accurate construction of edge maps.

Consistency becomes particularly desirable when computing structural properties of vector fields. Helman and Hesselink [10] compute a vector field's topological skeleton by segmenting the domain of the field using streamlines traced from each saddle of the field along its eigenvector directions. The nodes of the skeleton are critical points of the vector field and its arcs are the *separatrices* connecting them. Subsequently, the skeleton extraction has been extended to include periodic orbits [28]. Three dimensional variants of the topological skeleton have also been proposed [9, 12, 24, 27]. The reader should refer to [7, 14, 21] for more detailed
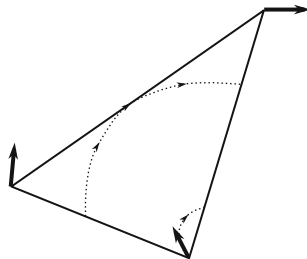
surveys. However, it is well known that computing the topological skeleton can be numerically unstable due to errors inherent in the integration of separatrices and inconsistencies among neighboring triangles [4, 8, 16, 23].

A number of techniques have been proposed to extract the topological skeleton in a stable and efficient manner [3, 17, 22, 28]. Recent work of Reininghaus and Hotz [18] construct a combinatorial vector based on Forman's discrete Morse theory [6]. Using combinatorial fields allows the extraction of a consistent topological structure. However, combinatorial vector fields are limited by their high complexity, leading to later improvements to the algorithm [19]. While provably consistent, it is unclear how close the topological structure of the combinatorial field is to that of the original, piecewise linear, field. By comparison, this work proposes a multi-resolution technique that is both consistent and provides control over the level of approximation of the field.

3 Edge Maps

Let $\mathbf{V} \colon \mathcal{M} \to \mathbb{R}^2$ be a 2-dimensional vector field defined on a manifold $\mathcal{M}$. $\mathbf{V}$ is represented as a set of vector values sampled on the vertices of a triangulation of $\mathcal{M}$. Specifically, each vertex p_i has the vector value $\mathbf{V}(p_i)$ associated with it. The vector values on the interior of each triangle T with vertices $\{p_i, p_j, p_k\}$ in the triangulation are interpolated linearly using $\mathbf{V}(p_i)$, $\mathbf{V}(p_j)$, and $\mathbf{V}(p_k)$ (see Fig. 1).

Given a vector field $\mathbf{V}$, we can define the *flow* $\phi(x, t)$ of $\mathbf{V}$. Treating $\mathbf{V}$ as a velocity field, the flow intuitively describes the parametric path that a massless particle travels according to the instantaneous velocity defined by $\mathbf{V}$. We define $\phi(x, t)$ as the solution of the differential equation:

$$\frac{d\phi(x, t)}{dt} = \mathbf{V}(x)$$

with the initial condition $\phi(x, 0) = x_0$. Fixing a point x and varying t, we call the collection of points produced by ϕ a *streamline*.

3.1 Preliminaries

For the remainder of this work, we will assume the vector values for **V** on the interior of a triangle are defined by linearly interpolating the three vectors at the vertices of a triangle, T. We make three simplifying assumptions: (1) the vectors at all the vertices of the triangle are non-zero, (2) the vectors at any two vertices sharing an edge are not antiparallel, and (3) the vectors at two vertices on an edge e are not both parallel to e. Any such configuration is unstable, and can be avoided by a slight perturbation. These small perturbations ensure a point to point mapping between the boundary of the triangle, and hence makes edge maps bijective. As we will see in the following sections, these assumptions will significantly reduce the complexity of many of the arguments as the locations of critical points become well defined.

Under the aforementioned assumptions, we can begin studying the properties of linearly varying vector fields. We first note the following two important properties that follow from our assumptions.

Property 1. Within T, **V** defines at most one critical point.

Proof. Critical points are defined as points x where $\mathbf{V}(x) = 0$. Consider the component scalar fields V_1, V_2 where $\mathbf{V}(x) = (V_1(x), V_2(x))$. As the triangle is linearly varying, the graphs of V_1 and V_2 are planes lifted from the triangle. Critical points are the set of points where both these two planes intersect the constant plane of height zero. As we know from planar geometry, two planes either intersect in another plane or a line, and consequently these three planes intersect in a plane, a line, or a point.

If these three planes intersect in a plane, this defines an infinite number of zeros, which can only happen when assumption (1) is violated. If these planes intersect in a line, the only way that line passes through T is when either (1) is violated (because two vertices are zero) or (2) is violated for at least two edges.

In all other cases, either a single critical point is defined (within the interior of T) or the field defines critical values outside of T. □

Property 2. On the line ℓ obtained by extending, from both sides, any edge e of a triangle T, there exists at the most one point where the vector field is tangential to ℓ.

Proof. Without loss of generality, rotate ℓ to be the x-axis and decompose **V** into its x and y components, i.e., $\mathbf{V}(\cdot) = (V_x(\cdot), V_y(\cdot))$. Following the linearity of the vector field **V**, V_y is also linear. By assumptions (1) and (3), $V_y = 0$ can only happen once, meaning that **V** is tangential to ℓ at only one point. We further note that on either side of this zero crossing, V_y has a different sign, and hence the flow changes directions from flowing upwards to downwards, or vice versa. □
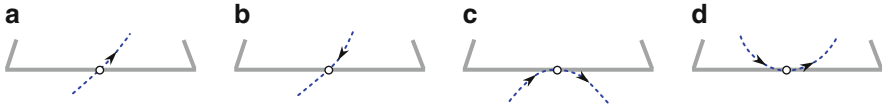
Fig. 2 Based on the flow at a point on the boundary of a triangle, the point can be classified as one of the following: (**a**) Inflow point, (**b**) Outflow point, (**c**) External transition point (ETP), and (**d**) Internal transition point (ITP)

3.2 Defining Edge Maps

For a triangle T, let ∂T and $\mathring{T}$ be the boundary and interior of T respectively. We first examine the behavior of the flow at any point on ∂T. The flow behavior can be classified into four types (as illustrated in Fig. 2) depending upon the location of that particle under flow $\phi(x, t)$ for positive and negative time t. Let $\varepsilon > 0$. Given a point $p \in \partial T$ such that $\phi(x, 0) = p$, we define the following:

Definition 1 (Inflow Point). If for all $t \in]0, \varepsilon]$, $\phi(x, t) \in \mathring{T}$ and $\phi(x, -t) \notin T$, then p is an *inflow point*.

Definition 2 (Outflow Point). If for all $t \in]0, \varepsilon]$, $\phi(x, t) \notin T$ and $\phi(x, -t) \in \mathring{T}$, then p is an *outflow point*.

Definition 3 (External Transition Point (ETP)). If for all $t \in]0, \varepsilon]$, $\phi(x, \pm t) \notin T$, then p is an *external transition point (ETP)*.

Definition 4 (Internal Transition Point (ITP)). If for all $t \in]0, \varepsilon]$, $\phi(x, \pm t) \in \mathring{T}$, then p is an *internal transition point (ITP)*.

The concept of transition points is similar to that of boundary switch points mentioned in [5, 15, 26]. The above four definitions account for all possible behaviors of flow as it touches ∂T, since our assumptions only allow the possibility where flow travels along the edge instantaneously. Our ultimate goal is to model the flow through $\mathring{T}$. To start, we first pair points as they travel along the same streamline.

Definition 5 (Origin-Destination Pair (o–d pair)). Let $a, b \in \mathbb{R}$ and $a \le b$ such that $\phi(x, [a, b]) \subset T$ where $p = \phi(x, a)$ and $q = \phi(x, b)$. We call (p, q) an origin-destination pair if the time interval $[a, b]$ is maximal where $p, q \in \partial T$ and $\phi(x,]a, b[) \subset \mathring{T}$. We call p an *origin point* and q a *destination point*.

Here, maximal interval means that the time range $[a, b]$ produces the largest possible streamline contained within $\mathring{T}$, bounded by two points p and q on ∂T. Inflow, outflow, and transition points all play different roles in o–d pairs. The simplest case is for inflow and outflow points. Since the streamline of an inflow point p flows to the $\mathring{T}$, p will be paired up with the point q (either an outflow point or ITP) where this streamline first touches the boundary after p.

An ITP flows to $\overset{\circ}{T}$ in both positive and negative time, thus creating at most two such maximal intervals for which a streamline is completely contained in $\overset{\circ}{T}$. Hence, an ITP may participate in two o–d pairs. In one pair it is an origin point while in the other it is a destination point. If the streamline is an orbit touching ∂T only at the ITP, we pair the ITP with itself, resulting in a single o–d pair. In this case, the streamline of the orbit has many time intervals $[a, b]$ where $\phi(x,]a, b[) \subset \overset{\circ}{T}$, all of which are of equal length. On the contrary, an ETP always forms an o–d pair with itself, since the streamline does not flow to $\overset{\circ}{T}$. Hence its maximal time range will be when $a = b$.

Thus, all origin points are either inflow points or transition points; and all destination points are either outflow points or transition points. Note that if x is chosen such that $\phi(x, t)$ is an orbit contained entirely in $\overset{\circ}{T}$, then such a streamline will never converge to ∂T and hence any p and q on it will not form an o–d pair. Moreover, there are some points that do not converge to ∂T, because of the presence of a critical point in T. Such points can be classified as *unmapped inflow points* (associated with a sink), *unmapped outflow points* (associated with a source), or *sepx points* (intersections of the saddle separatrices with ∂T), and they are not included in the definition of o–d pairs.

Since we can determine the existence of an o–d pair for every point on ∂T we can define a mapping to describe the transversal behavior of flow through points on ∂T. Let $P \subset \partial T$ be the set of all origin points, and $Q \subset \partial T$ be the set of all destination points.

Definition 6 (Edge Map). An *edge map* of T is defined as a map $\xi \colon P \to Q$, such that $\xi(p) = q$, if (p, q) is an o–d pair.

We will also call q the *image* of p under ξ.

We claim that ξ is a bijection between P and Q, and thus its inverse is also well defined. We call ξ as the *forward edge map*, since it maps an origin point to its destination point under forward flow. Its inverse ξ^{-1} maps a destination point to its origin point, and hence is called the *backward edge map*.

Using the forward and backward edge maps, integration of streamlines can be replaced with a map lookup that traverses the triangle. Using only this lookup, a streamline travelling through a triangulated vector field is approximated as a sequence of points on the boundaries of triangles through the field. This approximation discards the flow behavior of the interior of triangles, but as we shall see, maintains enough information to maintain consistency of streamlines.

3.3 Approximating Edge Maps

In this paper, we intend to define a feasible representation for storing and using an *approximated edge map* ξ^* as a data structure to represent ξ. We achieve this by grouping the point to point exact mapping ξ into connected intervals which preserve its ordering, forming what we call the *links*.

Definition 7 (Link). Let O be a connected subset of P, and D be a connected subset of Q, where $D = \xi(O)$. Let ∂O be the end points (boundary) of O. We call $\zeta: O \rightarrow D$ a *link* if ζ is continuous, order-preserving, and $\zeta(x) = \xi(x)$ for each $x \in \partial O$. We call the set O an *origin interval* and the set D a *destination interval*.

A link ζ allows for a second level of approximation of ξ on some subsets of the domain of ξ. Let $\overset{\circ}{O}$ be the interior of O. For each point $p \in \overset{\circ}{O}$, we allow the mapped points $\zeta(p)$ to shift from its original $\xi(p)$, except at the boundaries of O.

Although, ζ is an approximation, we at least require that it preserves the ordering of streamlines. The property of *order-preservation* handles this notion by disallowing links which have streamlines that cross. To enforce order-preservation, first note that since O is a connected subset of ∂T, it can be parameterized as a single interval. Similarly, since ζ is continuous, we will only build links where $\xi(O)$ is a single interval, again parameterizable. As long as we preserve the ordering within this parameterization, ζ is a valid link. Thus, the property of order-preservation leads to the following theorem.

Theorem 1. *Order-preserving links always produce consistent (pairwise disjoint) streamlines.*

Since each ζ may only be defined on a subset of P, to completely approximate ξ we need a set of links. This motivates the following definition.

Definition 8 (Approximated Edge Map). An *approximated edge map* ξ^* of ξ is a collection of n links $\zeta_i : O_i \rightarrow D_i$, such that the $\{O_i\}$ and $\{D_i\}$ form partitioning of P and Q respectively.

We can see that an edge map can be subdivided into links in many ways to form such a partitioning. Since the boundaries of each link are "snapped" to require $\zeta(x) = \xi(x)$, using more links enforces more accuracy. Once the partitioning of P is defined, one way of approximating ξ as ξ^* is creating links ζ_i which are linearly mapped between O_i and D_i, as proposed in [1]. Such a ζ satisfies the properties of order-preservation, and provides a simpler map lookup computation.

Earlier, we pointed out that certain points on ∂T can be classified as unmapped inflow, unmapped outflow or sepx points, and do not participate in any o–d pair. It follows from the definition of the link that these points cannot be included in either of the origin or destination intervals, and hence do not get included in any link. Connected subsets of such points are called *unmapped intervals*.

3.4 Base Edge Maps

Links can be created by *merging* the origin and destination sets of adjacent o–d pairs such that the origin and destination intervals remain connected sets and the orientation of the link is preserved. It turns out that this merging process can be expanded only so much, until specific points where the continuity of the flow

Fig. 3 The *base edge map*
for the triangle in Fig. 1 has
three links (shown in different
colors). It is generated by
splitting the boundary at
ITP's (*white circles*), their
images (*grey dots*), and ETP's
(*white squares*)
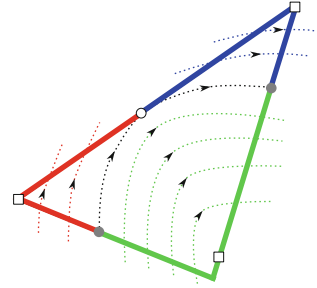
breaks. At this maximal amount of merging, we have a minimal set of links in the
approximated edge map.

Definition 9 (Base Edge Map). A *base edge map* of T is an approximated edge
map ξ^* such that the number of links is minimal. Links of a base map are called
base links.

This merging of o–d pairs into a single link cannot be done across transition
points, sepx points, and ITP images. This follows since orientation of a link cannot
be preserved across transition points while sepx points and ITP images break the
continuity of the map. Thus, the intervals in a base edge map are bounded by the
transition points, sepx points, and the image points of ITPs. A base edge map can
be readily constructed by identifying these points and splitting the boundary of
a triangle at these points into intervals. The intervals can be paired up into links
using connectivity information. Figure 3 shows the creation of base edge maps for
a regular triangle (with no critical point in the interior).

4   Classification of Base Edge Maps

There are many ways to approximate an edge map ξ as a set of links. Using the
concept of a base edge map, many different edge maps can be reduced to the
same base edge map. Thus, a base edge map can be thought of as representative
of an infinite number of edge maps, each representing the same flow behavior.
This motivates a study of all possible base edge maps, which leads to a notion of
equivalence classes of the edge maps that represent all possible types of linearly
varying flow. In what follows, all proofs are included in the appendix.

4.1   *Equivalence of Maps*

While the notion of reducing any edge map to its base edge map is well defined,
we need to define why two base edge maps are considered equivalent in order to
declare them as belonging to the same class. Before discussing equivalence, we shall

enumerate all the features of a map that we would like to include when comparing two maps.

- *Pairing* of intervals (O_i, D_i) as links.
- *Orientation* of links $(O_i \rightarrow D_i)$.
- *Cyclic order* of all intervals (including unmapped) on ∂T.

We define a way to construct a mixed graph from any edge map ξ approximated by ξ^* and the set of links $\{\zeta_1, \zeta_2, \ldots, \zeta_n\}$. Our goal is to capture all the above features into the graph. A **mixed graph** $\mathscr{G}(V, E, A)$ can be constructed such that:

- Nodes: $V = \{I_k\}$ where I_k is either an origin interval O_i, a destination interval D_i, an unmapped interval U_j, or a sink or source S_p in $\mathring{T}$.
- Undirected Edges: $E = \{\{I_1, I_2\}\}$ where I_1 and I_2 are adjacent on ∂T.
- Directed Edges: $A = \{(I_1, I_2)\}$ where either $\zeta_i : I_1 \rightarrow I_2$ (corresponding to a link); $I_1 = U_j$ and $I_2 = S_p$ (an unmapped inflow interval); or $I_1 = S_p$ and $I_2 = U_j$ (an unmapped outflow interval).

An example of such a graph is shown in Fig. 4. Though unmapped intervals do not participate in the edge map, they have a structural bearing on the map and consequently on the mixed graph. They separate links that would otherwise appear contiguous and could be potentially merged. In the case of a saddle, while the sepx points are also unmapped, they need not have an explicit place in the graph
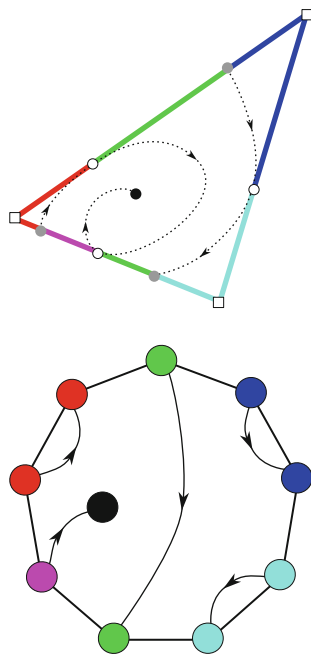
Fig. 4 Construction of mixed graph from a map. Points of the magenta interval do not form o–d pairs since they flow directly to S_p

as their existence can be inferred from its neighboring intervals (and consequently the existence of a saddle in $\overset{\circ}{T}$).

Definition 10 (Equivalence). Two edge maps are considered equivalent if the mixed graphs of their base edge maps are *isomorphic*.

Under this definition of equivalence, we present a group theoretic approach for counting all possible equivalence classes of base edge maps. We use the Groups, Algorithms, and Programming (GAP) system [20] to do the following computation. There are three important steps: *generation* of initial sample space; *restricting* this space to those possible in linearly varying flow; and identifying *equivalences*.

4.2   Initial Sample Space

Our analysis for all possible equivalence classes first considers all possibilities for links and unmapped intervals. We start by disallowing an interval from spanning beyond an edge. This simplifying assumption helps enumerate the sample space, and later we shall remove these splits caused by vertices, allowing links to span across them. From the definition of base links and given the property that all links of a base edge map are bounded by transition points, sepx points, and ITP images, any base edge map has a bounded number of links and unmapped intervals.

Let a, b, c symbolically represent the edges of T and d be either the sink or source in $\overset{\circ}{T}$, if it exists. For an interval I, its *location*, $loc(I)$, can be any of the four values in $L = \{a, b, c, d\}$. In the following lemma we show that in $\mathscr{G}$, no two directed edges having their origin nodes in the same location can have their destination nodes in the same location. Said more formally:

Lemma 1 (Edge Level Links). *Let $\mathscr{G}(V, E, A)$ be a mixed graph of an edge map where no node I_i spans more than one edge of ∂T and let $loc(I_i)$ be its location. Consider two directed edges $(I_1, I_2), (I_3, I_4) \in A$. If $loc(I_1) = loc(I_3)$ and $loc(I_2) = loc(I_4)$, then $(I_1, I_2), (I_3, I_4)$ can be merged.*

From Lemma 1 we see that there are a maximum of 13 directed edges in the graph of any edge map with edge level links, as each pair in $S = L \times L$ can only appear once and the pair (d, d) cannot appear because d is either a source or a sink, but not both. Moreover, no pair $(d, \cdot)$ can co-exist with a pair $(\cdot, d)$, again because d is either a source or a sink. Thus, each configuration for any set of directed edges A can be represented as certain elements of $\mathscr{P}(S)$, the power set of S. Consequently, the number of possible configurations for directed edges is 2^{13}.

Under rotations of T, certain directed edges are equivalent. For example, $\{(a, b), (b, c)\}$ is equivalent to $\{(b, c), (c, a)\}$. Similarly, $\{(b, b), (c, c)\}$ is equivalent to $\{(c, c), (a, a)\}$ and $\{(a, a), (b, b)\}$. We next eliminate such equivalences under rotation by using the action of a permutation group that imparts rotations $(a \rightarrow b \rightarrow c \rightarrow a)$ and $(a \rightarrow c \rightarrow b \rightarrow a)$.

These permutations allow us to enumerate the possibilities for the set A; however, they still lack any cyclic ordering of nodes to form the undirected edges E in $\mathscr{G}$. We generate all the mixed graphs under the constraint that there is a maximum of one TP on the interior of an edge (a consequence of Property 2) by grouping all intervals sharing the same edge label. We do this by first generating permutations of all origin/unmapped inflow nodes and destination/unmapped outflow nodes separately on every edge. All possible orderings of nodes on that edge are then the Cartesian product of these two sets. This ultimately constructs the set of all possible mixed graphs (our initial sample space) which numbers more than 100 million.

4.3 Restricting Cases

Our initial enumeration of all graphs is quite large, but contains many invalid graphs. Using the following lemmas based on the linear nature of $\mathbf{V}$, we filter this initial set. In particular, the following must hold:

Lemma 2 (ITP-vertex). *An ITP cannot exist on a vertex of T.*

Next, Lemma 3 states an important fact that can be used to qualify a critical point based on the existence of an ETP on the interior of an edge of the triangle.

Lemma 3 (ETP-Saddle). *Let $q \in \partial T - \{p_i, p_j, p_k\}$ be an ETP. If there exists a critical point $S_p \in \mathring{T}$, then S_p is a saddle.*

Based on these lemmas, we enforce the following rules on all mixed graphs:

1. Directed edges of $\mathscr{G}$ cannot intersect. (Since streamlines never intersecting in piecewise linear flow.)
2. ITP cannot exist on vertices. (Lemma 2)
3. ETPs cannot exist on edges when there is source or sink (Lemma 3 and Property 1).

We can label some undirected edges of a mixed graph as either ITPs or ETPs using the combinatorial structure of the graph. For an undirected edge $\{I_1, I_2\}$ in $\mathscr{G}$, if I_1 and I_2 switch between inflow to outflow, then $\{I_1, I_2\}$ is a *combinatorial TP*. Moreover, if I_1 and I_2 do not have a directed edge between them in A, then the TP is an *combinatorial ITP*. Images of an ITP are directly implied by its directed edges. Similarly, if I_1 and I_2 have a directed edge between them, then the TP is a *combinatorial ETP*. We remark that in a very few number of cases, combinatorial ETPs actually correspond to an interior flow that wraps to itself, creating an orbit in the triangle. These cases are not distinguished by the flow maps, but can only appear if there is no source or sink within $\mathring{T}$.

Though the above analysis does not label all the undirected edges, these are sufficient to enforce the above mentioned rules to restrict the sample space of mixed graphs to under one thousand. After labeling combinatorial ITPs and ETPs, the

second and third rule can be enforced directly. We enforce the first rule using a stack method of detecting intersections between links. In a valid map, any link that has its intervals on ∂T divides the triangle into two halves such that no other link has intervals in both the halves, thus forming a last in, first out sequence on ∂T equivalent to pairing parentheses in a mathematical expression.

4.4 Identifying Equivalences

For the final equivalence computation in GAP, we merge contiguous directed edges so that all mixed graphs become base edge maps. We next use the dihedral group for all graph symmetries as well as the permutation group for all labelings of nodes. The dihedral group is useful for comparing the mixed graphs under all rigid transformations. The permutation group also includes inversion of the flow direction of directed edges to compare similarities in base edge maps under global inversion of flow. Using GAP, we enumerate all graphs that fall in the same orbit under its actions, each set of which represents an equivalence class for the graphs. This process produces 43 map classes as an output.

Since our mixed graph definition does not explicitly encode saddles or orbits, we then manually invalidate more cases based on some additional lemmas, many of which would be challenging to encode within GAP, but are simple to remove by inspection. We rely on two properties for the behavior of ITPs:

Lemma 4 (ITP-Saddle). *If there is a saddle in $\overset{\circ}{T}$, then an ITP cannot be present on ∂T.*

Additionally, Lemma 5 limits the location of a critical point S_p within the interior of the triangle when an ITP exists. In particular, S_p must be enclosed by the streamline extending from an ITP.

Lemma 5 (ITP-CP Enclosure). *Let $x \in \partial T$ be an ITP whose images $x_b, x_f \in \partial T$ and S_p be a critical point in $\overset{\circ}{T}$. S_p is an element of the area bounded by the streamline $\phi(x, t)$ and ∂T where $t \in [a, b]$ s.t. $\phi(x, a) = x_b$ and $\phi(x, b) = x_f$.*

Thus, the following rules are enforced on the remaining cases to get the final equivalence classes:

1. The graphs should not violate Lemma 4.
2. The graphs should not violate Lemma 5.
3. There can be a maximum of one critical point. If there is a sink or a source in the map, there cannot a sepx point (Property 1, since a sepx point means there must additionally be a saddle).
4. There have to be exactly four sepx points or none at all (By definition of a linear saddle).

To enforce the above rules, we need to identify *combinatorial sepx* points in the graphs. Since, we have merged all contiguous directed edges, the graphs represent the base edge maps. Thus, any unlabeled undirected edges have to correspond to sepx points, since all ETP, ITP and ITP images have already been labeled. Knowing the labeling of each undirected edge allows us to hand invalidate 20 of the 43 cases produced by GAP. 23 classes persist after the invalidations, visualized in Fig. 5. We have generated vector values for triangles to realize each of these cases as well, confirming the following result:

Theorem 2. *Under Definition 10 for equivalence, there are 23 equivalence classes of edge maps for triangles in a linearly varying vector field.*

5 Discussion

Edge maps are a natural way to encode one facet of the information a triangulated vector field presents. The study of the equivalence classes of edge maps helps us understand different structures of flow possible on a triangle. Every class represents a large number of possible realizations of the flow. While some of the edge map classes represent a more generic flow in terms of a point-to-point map, some of them may reflect degeneracies in the flow. For example, class 4 (first column, third row in Fig. 5) represents the flow imparted by an orbit in a triangle. While it is widely known that such a field is unstable under small perturbations, its corresponding edge map class also represents this instability.

The main benefit of edge maps is that they explicitly store the origins and destinations of flow through individual triangles. This gives a direct control on how the streamlines are computed, enabling us to enforce consistency. While they store how streamlines travel across triangles, they discard any notion of what is happening within the triangle. However, they can still preserve the global structure of the flow in the sense that starting at a single point and traversing through a series of maps will give an accurate representation of the destination even if the path it takes is only approximated. If a higher resolution of the shape of a streamline is desired, it may be achieved by using numerical integration as a progressive computation. This is generally not required if the mesh is sufficiently fine.

In their current form, the maps drop all notion of time in their construction. One could model a "unit" of time as a jump across a single triangle. It might be more meaningful to add how long it takes to get from origin to destination across the triangle. The time it takes to travel through a link could then be added into the edge map. This modification would allow a more accurate representation of flow.

We believe that edge maps have the potential to complement existing techniques for processing vector fields. Since they maintain consistency while providing a level of control over accuracy, one could conceivably design algorithms for visualization and topological analysis that can leverage these properties. Some example applications we have already considered include error analysis of streamline integration techniques [1].
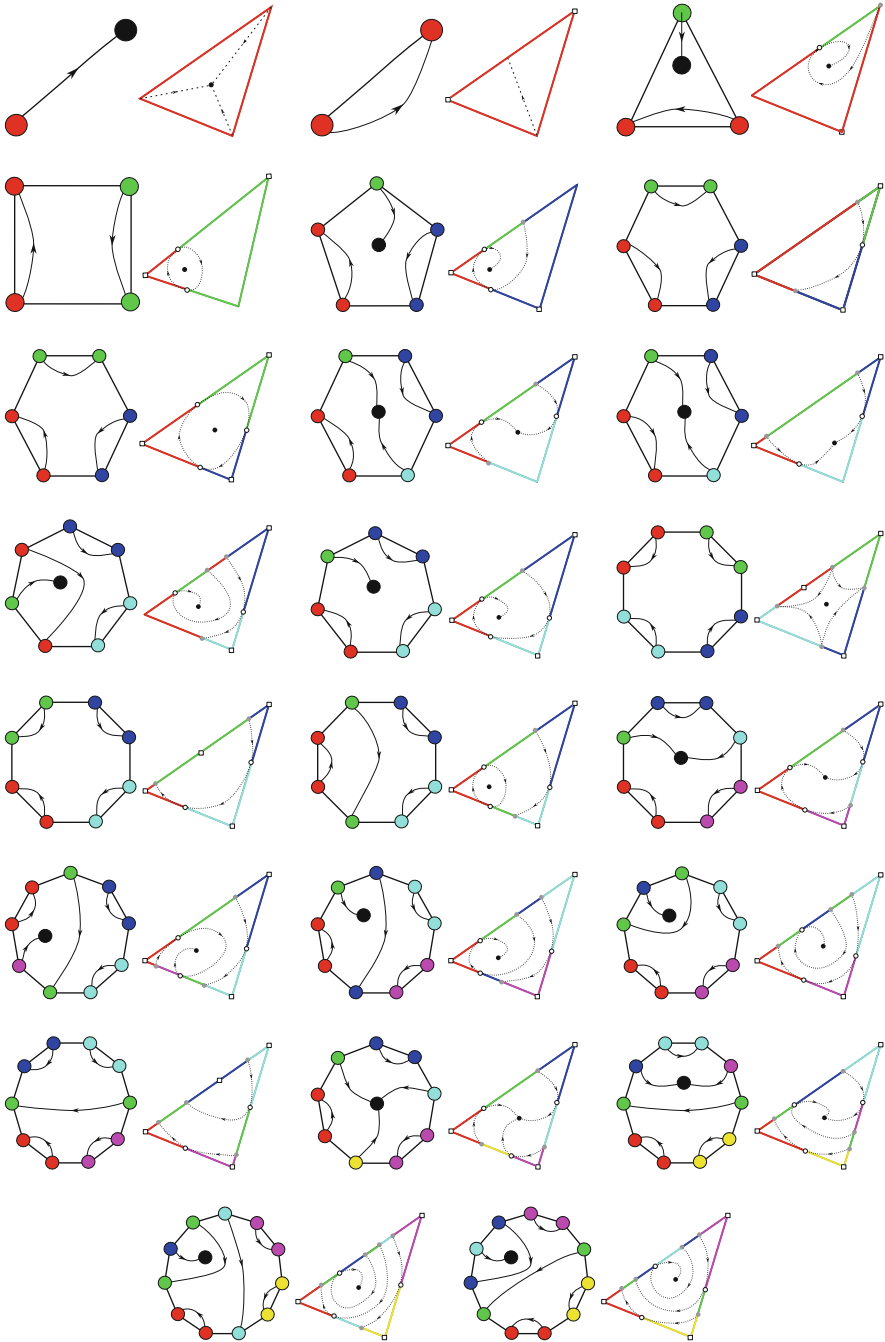
Fig. 5 The 23 equivalent classes of mixed graphs, along with one possible rendition of edge map for piecewise linear flow for each class, in the order of increasing number of links

The structure of an edge map depends on the flow as well as the geometry of the underlying mesh. The current study of edge maps is based on linear interpolation. As future work, there are two apparent ways to generalize this study. One is by extending it to a generalized interpolation scheme; and another way is to extend it to higher dimensional simplices and other types of meshes.

Appendix

Proofs from Sect. 4.2

Proof (Lemma 1). In all the cases that follow, if I_1 and I_3 lie on an edge of T then we assume that there exists a subset $I_5 \in \partial T$ such that I_5 lies between I_1 and I_3. Similarly, there exists $I_6 \in \partial T$ between I_2 and I_4. In such a scenario, I_5 is a set of inflow points and I_6 is a set of outflow points, otherwise there will be more than one TP on the same edge. We know from Property 2 that this is not possible. Let a, b, c be the edges of T and d the critical point in $\mathring{T}$. We enumerate all possibilities of locations for the four intervals:

1. $loc(I_1)$ and $loc(I_3)$ is any edge, e; and $loc(I_2)$ and $loc(I_4)$ is d.
2. $loc(I_1)$ and $loc(I_3)$ is d; and $loc(I_2)$ and $loc(I_4)$ is any edge, e.
   (This is symmetric to case 1 in reverse flow.)
3. $loc(I_1)$ and $loc(I_3)$ is any edge, e_1; and $loc(I_2)$ and $loc(I_4)$ is another edge, e_2.
4. $loc(I_1)$ and $loc(I_3)$ is any edge, e; and $loc(I_2)$ and $loc(I_4)$ is the same edge e.
5. $loc(I_1), loc(I_3), loc(I_2),$ and $loc(I_4)$ is d.
   (This case is not possible since the one critical point is either source or sink but not both.)

From the discussion in Sect. 3.4 we know that we can merge adjacent o–d pairs except across transition points, sepx points, and ITP images. For all the above cases, streamlines on I_5 and I_6 are bounded by (I_1, I_2) and (I_3, I_4) irrespective of their locations. Therefore, since there cannot be a TP on I_5 or I_6, we can also rule out ITP images. Similarly, sepx points are impossible since all four sepx points will have to lie on I_5 and I_6 which would imply a TP on them. Thus, under the given constraint, (I_1, I_2) and (I_3, I_4) can be merged.   □

Proofs from Sect. 4.3

Proof (Lemma 2). Let vertex p_j of T be an ITP. From the definition of ITP, the curve $\phi(x, \pm t) \in \mathring{T}$ for all $t \in]0, \varepsilon]$, such that $\phi(x, 0) = p_j$. The streamline $\phi(x, \pm t)$ is a C^1 continuous curve, while the two edges of T meeting at p_j define a
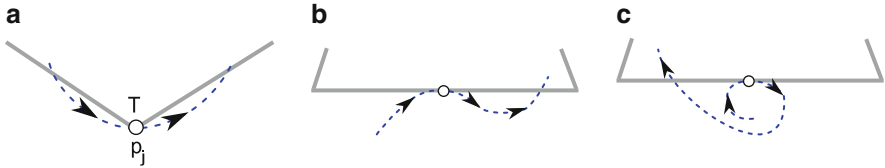
Fig. 6 (**a**) In Lemma 2, a streamline passing through a vertex (p_j) of triangle T must exit and re-enter T. (**b**) The first case in Lemma 3, multiple switches in flow across line of the edge. (**c**) The second case in Lemma 3, more than one critical point

corner. By C^1 continuity, $\phi(x, \pm t)$ has a well defined tangent everywhere, including at p_j, but since the interior angle of T at p_j is less than π, this is not possible for $\phi(x, \pm t)$ being contained in $\mathring{T}$. Thus, the streamline $\phi(x, \pm t)$ must exit T to maintain its continuity (as shown in Fig. 6a). This contradicts the definition of an ITP, implying an ITP is not possible on a vertex. □

Proof (Lemma 3). Let an ETP x_e exist on an edge a, of T. Thus, $\phi(x_e, \pm t) \notin T$ for $t \in]0, \varepsilon]$. Let $S_p \in \mathring{T}$ be the critical point. From Property 1 we know that there is exactly one interior critical point in the vector field. Therefore, if S_p is a sink, all streamlines eventually flow to it; if S_p is a source, all streamlines emerge from it; and if S_p is an orbit, all streamlines are closed loops. Thus, if S_p is a sink or a source, $\phi(x_e, t)$ will have to flow to, or emerge from it. This cannot happen without crossing the line ℓ of edge a. There are two ways in which $\phi(x_e, t)$ can cross ℓ. First, it may make multiple switches of direction (Fig. 6b) over ℓ, violating Property 2. Second, one of the images (backward or forward) gets trapped in the enclosure of $\phi(x_e, t)$ and flows to a critical point inside (Fig. 6c), violating the fact that there is exactly one defined critical point. If S_p is an orbit, the closed loop has to enclose S_p else it would define another orbit. Thus again $\phi(x_e, t)$ is forced to cross ℓ, violating Property 2. Thus S_p has to be a saddle. □

Proofs from Sect. 4.4

Proof (Lemma 4). Let an ITP x_i exist on an edge a of T, Thus, $\phi(x_i, \pm t) \in \mathring{T}$ for $t \in]0, \varepsilon]$. Let $S_p \in \mathring{T}$ be the critical point. Assume, to build a contradiction, that S_p is a saddle. This assumption implies that all streamlines including $\phi(x_i, t)$ are parallel to the separatrices of the saddle as $t \to \pm\infty$. We show that $\phi(x_i, t)$ crosses the line of an edge of T multiple times to stay parallel to the separatrices in those limits of t, contradicting Property 2. This follows since at least one separatrix intersects the line of every edge of T forcing multiple intersections of $\phi(x_i, t)$ with the line of the edge to which x_i belongs. Hence, S_p cannot be a saddle. The cases for saddle sectors are shown in Fig. 7a, b. □

Proof (Lemma 5). Let an ITP x_i exist on an edge a, of T and S_p be a critical point in $\mathring{T}$. Assume, to build a contradiction, that S_p exists outside the enclosure of
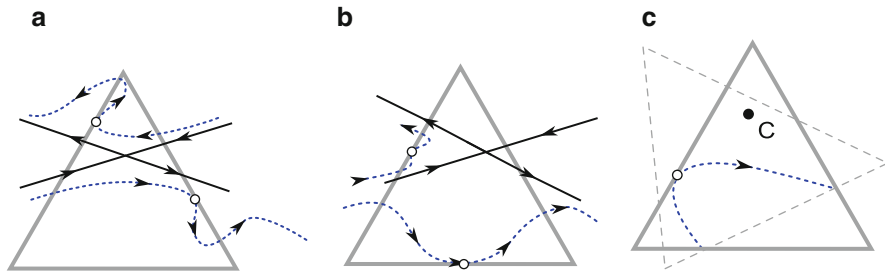
Fig. 7 (**a**) and (**b**) Lemma 4: Cases of saddle sectors intersecting ∂T invalidating the existence of an ITP on ∂T. (**c**) Construction of a new triangle as in Lemma 5 to place both the forward and backward images of an ITP are on the same edge

$\phi(x_i, t)$ in $\overset{\circ}{T}$. We show that this is not possible. Since x_i is an ITP, S_p is not a saddle according to Lemma 4. Let a, b, c be the three edges of T. Now we enumerate all possible combinations of edges on which x_i, x_f, and x_b can lie:

1. $\text{edge}(x_i) = \text{edge}(x_b) = \text{edge}(x_f)$.
   The only way to realize this case is an orbit such that $x_i = x_f = x_b$. Such an orbit immediately encloses the critical point.
2. $\text{edge}(x_i) = \text{edge}(x_b)$ or $\text{edge}(x_i) = \text{edge}(x_f)$.
   These two cases are symmetrical under reversal of flow. If either x_f or x_b lie on the same edge as x_i, by Property 2, the other image gets enclosed into the $\phi(x_i, t)$ and has to flow to a critical point.
3. $\text{edge}(x_i) \neq \text{edge}(x_b) = \text{edge}(x_f)$.
   In this case, both x_b and x_f lie on the same edge indicating a switch from inflow to outflow. This implies that there exists at least one TP between them including the two points. If one of x_f and x_b is a TP, it has to be an ITP since part of its streamline is already in $\overset{\circ}{T}$. Due to Property 2, the other image of this ITP has to flow into a critical point in the enclosure of $\phi(x_i, t)$. If a point in the interior is a TP, this TP cannot be an ETP since we have already established that S_p is not a saddle and an ETP would violate Lemma 3. Now one of the images of this ITP can flow back to the same edge, but the other has to flow to S_p. Thus S_p is in the enclosure of $\phi(x_i, t)$.
4. $\text{edge}(x_i) \neq \text{edge}(x_b) \neq \text{edge}(x_f)$.
   In this case, we can always construct a new triangle by connecting x_f and x_b as an edge and arbitrarily choosing a third point such that S_p still lies in the interior of the new triangle as shown in Fig. 7c. An argument identical to case 3 can now be applied. □

References

1. Bhatia, H., Jadhav, S., Bremer, P.-T., Chen, G., Levine, J. A., Nonato, L. G., Pascucci, V.: Edge maps: Representing flow with bounded error. In: Pacific Visualization Symposium (PacificVis), 2011 IEEE, pp. 75–82, March 2011

2. Cabral, B., Leedom, L.C.: Imaging vector fields using line integral convolution. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, Anaheim, CA, pp. 263–270. ACM, New York (1993)

3. Chen, G., Mischaikow, K., Laramee, R.S., Pilarczyk, P., Zhang, E.: Vector field editing and periodic orbit extraction using Morse decomposition. IEEE Trans. Vis. Comput. Graph. **13**(4), 769–785 (2007)

4. Chen, G., Mischaikow, K., Laramee, R.S., Zhang, E.: Efficient morse decompositions of vector fields. IEEE Trans. Vis. Comput. Graph. **14**(4), 848–862 (2008)

5. de Leeuw, W., van Liere, R.: Collapsing flow topology using area metrics. In: Proceedings of the Conference on Visualization '99: celebrating ten years, VIS '99, San Francisco, CA, pp. 349–354. IEEE Computer Society Press, Los Alamitos (1999)

6. Forman, R.: A user's guide to discrete Morse theory. In: Proc. of the 2001 Internat. Conf. on Formal Power Series and Algebraic Combinatorics, A special volume of Advances in Applied Mathematics, p. 48 (2001)

7. Garth, C., Tricoche, X.: Topology- and feature-based flow visualization: Methods and applications. In: SIAM Conference on Geometric Design and Computing (2005)

8. Garth, C., Krishnan, H., Tricoche, X., Tricoche, T., Joy, K.I.: Generation of accurate integral surfaces in time-dependent vector fields. IEEE Trans. Vis. Comput. Graph. **14**(6), 1404–1411 (2008)

9. Globus, A., Levit, C. Lasinski, T.: A tool for visualizing the topology of three-dimensional vector fields. In: Proceedings of the 2nd Conference on Visualization '91, VIS '91, San Diego, CA, pp. 33–40. IEEE Computer Society Press, Los Alamitos (1991)

10. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluidflow data sets. IEEE Comput. **22**(8), 27–36 (1989)

11. Hirsch, M.W., Smale, S., Devaney, R.L.: Differential Equations, Dynamical Systems, and an Introduction to Chaos, 2nd edn. Elsevier, Amsterdam (2004)

12. Janine, K.M., Bennett, J., Scheuermann, G., Hamann, B., Joy, K.I.: Topological segmentation in three-dimensional vector fields. IEEE Trans. Visual. Comput. Graph. **10**, 198–205 (2004)

13. Kipfer, P., Reck, F., Greiner, G.: Local exact particle tracing on unstructured grids. Comput. Graph. Forum **22**, 133–142 (2003)

14. Laramee, R.S., Hauser, H., Zhao, L., Post, F.H.: Topology based flow visualization: The state of the art. In: Topology-Based Methods in Visualization (Proc. Topo-in-Vis 2005), Mathematics and Visualization, pp. 1–19. Springer, Berlin (2007)

15. Lodha, S.K., Renteria, J.C., Roskin, K.M.: Topology preserving compression of 2d vector fields. In: Proceedings of the conference on Visualization '00, VIS '00, pp. 343–350. IEEE Computer Society Press, Los Alamitos, CA, USA (2000)

16. Nielson, G.M., Jung, I.-H.: Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. IEEE Trans. Vis. Comput. Graph. **5**(4), 360–372 (1999)

17. Polthier, K., Preuß, E.: Identifying vector fields singularities using a discrete Hodge decomposition. In: Hege, H.C., Polthier, K. (eds.) Mathematical Visualization III, pp. 112–134 (2003)

18. Reininghaus, J., Hotz, I.: Combinatorial 2d vector field topology extraction and simplification. In: Pascucci, V., Tricoche, X., Hagen, H., Tierny, J. (eds.) Topological Methods in Data Analysis and Visualization. Theory, Algorithms, and Applications. (TopoInVis'09), pp. 103–114. Springer, Berlin (2009)

19. Reininghaus, J., Löwen, C., Hotz, I.: Fast combinatorial vector field topology. IEEE Trans. Visual. Comput. Graph. **17**(10), 1433–1443 (2010). doi:10.1109/TVCG.2010.235

20. Schönert, M., et al.: GAP – Groups, Algorithms, and Programming. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, 5th edn. Aachen, Germany (1995)

21. Scheuermann, G., Tricoche, X.: Topological methods in flow visualization. In: Hansen, C., Johnson, C. (eds.) Visualization Handbook, pp. 341–356. Elsevier, Amsterdam (2004)

22. Scheuermann, G., Krüger, H., Menzel, M., Rockwood, A.P.: Visualizing nonlinear vector field topology. IEEE Trans. Visual. Comput. Graph. **4**(2), 109–116 (1998)

23. Scheuermann, G., Bobach, T., Hagen, H., Mahrous, K., Hamann, B., Joy, K.I., Kollmann, W.: A tetrahedra-based stream surface algorithm. In: VIS '01: Proceedings of the conference on Visualization '01, pp. 151–158. IEEE Computer Society, Washington, DC, USA (2001)
24. Theisel, H., Weinkauf, T., Hege, H.-C., Seidel, H.-P.: On the applicability of topological methods for complex flow data. In: Hauser, H., Hagen, H., Theisel, H. (eds.) Topology-based Methods in Visualization, Mathematics and Visualization, pp. 105–120. Springer, Berlin (2007)
25. Turk, G., Banks, D.: Image-guided streamline placement. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, SIGGRAPH '96, pp. 453–460. ACM, New York (1996)
26. Weinkauf, T., Theisel, H., Hege, H.-C., Seidel, H.-P.: Boundary switch connectors for topological visualization of complex 3D vector fields. In: Deussen, O., Hansen, C., Keim, D., Saupe, D. (eds.) Symposium on Visualization, pp. 183–192. Eurographics Association, Konstanz (2004)
27. Weinkauf, T., Theisel, H., Shi, K., Hege, H.-C., Seidel, H.P.: Extracting higher order critical points and topological simplification of 3D vector fields. In: IEEE Visualization, 2005. VIS 05. pp. 559–566 (2005)
28. Wischgoll, T., Scheuermann, G.: Detection and visualization of closed streamlines in planar fields. IEEE Trans. Visual. Comput. Graph. **7**(2), 165–172 (2001)

Cusps of Characteristic Curves
and Intersection-Aware Visualization
of Path and Streak Lines

Tino Weinkauf, Holger Theisel, and Olga Sorkine

1 Introduction

Dynamic flow phenomena play an important role in many applications. In this paper, we are particularly interested in 2D time-dependent flows. Their three-dimensional space-time domain allows essentially for the following visualization options:

- *Static space-time visualization*: Time is explicitly incorporated as the third dimension and the geometry or texture to be visualized is shown in the space-time volume. Examples are tracked critical points shown as line-type structures in space-time [11, 12], or path lines stretching through the volume (cf. Fig. 2b).
- *Dynamic visualization in space*: The temporal behavior is captured in an animation of a 2D visualization. Examples are particle animations [4], animations of FTLE color plots [2], animated texture-based flow visualizations [15, 17], and many more.
- *Static visualization in space*: The temporal behavior is encoded in the 2D visualization by means of color or other graphical attributes. An example is the visualization of path lines where time can be mapped to the width of the line. An example for 3D flows is given by Wiebel and Scheuermann [16], where bundles of path and streak lines running through the same spatial location are visualized in a static image.

T. Weinkauf (✉) · O. Sorkine
Courant Institute of Mathematical Sciences, New York University, 715 Broadway, New York, NY 10003, USA
e-mail: weinkauf@courant.nyu.edu; sorkine@courant.nyu.edu

H. Theisel
AG Visual Computing, Otto-von-Guericke-Universität, Universitätsplatz 2, 39106 Magdeburg, Germany
e-mail: theisel@isg.cs.uni-magdeburg.de

Static visualizations not only have the advantage of being perfectly suited for non-dynamic types of media such as paper, but they also compile all information into a single image, whereas animations break it into several transient impressions. Clearly, both the dynamic and the static approach have their merits.

In this paper we are concerned with static visualizations of characteristic curves (stream, path, streak, time lines; Sect. 2). While visualizing them in 3D space-time is always a valid option, note that this boils down to a 2D projection anyway once a snapshot is taken for purposes such as printing, and the non-orthogonal, perspective projection inevitably introduces distortions as showcased throughout the paper. In contrast, visualizing characteristic curves in the spatial domain (by means of an orthogonal, orthographic projection) leaves the physically meaningful spatial domain undistorted and the temporal dimension can be encoded in graphical attributes.

Various stream line placement methods are available that aim at expressive visualizations by distributing stream lines (of a time step or a steady flow) in an evenly-spaced manner [3, 6, 8, 13], or for 3D steady flows in a view-dependent fashion to reduce visual clutter [5]. To the best of our knowledge and much to our surprise, there seem to be no existing methods for the placement of the other three types of characteristic curves in time-dependent flows. There are many texture- or geometry-based approaches for path, streak, and time lines in general [1, 9, 15], but none of them take care of the distances between lines or possible intersections.

In this paper, we develop a method to create uncluttered, static visualizations of path and streak lines. To that end, we first analyze the challenges that come with the purely spatial depiction of these curves. Besides their obvious intersections in space, we pay special attention to cusps (isolated points on the curve with abruptly turning tangent direction) and self-intersections (Sects. 3 and 4). It turns out that both phenomena are closely related to the occurrence of critical points in the original flow. Based on the gained insight into the intricacies of visualizing path and streak lines in space, we design a placement algorithm for them that makes use of the flow topology and aims at cusp-free and intersection-aware visualizations (Sect. 5). As we will show in our results (Sect. 6), an *entirely* intersection-free visualization is of limited use for turbulent flows. We therefore allow for a small number of (self-)intersecting path or streak lines.

Notation: We consider a 2D time-dependent vector field $\mathbf{v}(\mathbf{x}, t)$ over the domain $D \times T$, where $D \subseteq \mathbb{R}^2$ is the spatial domain and T is a time interval. We write $(2 + 1)$-dimensional variables with a bar like $\bar{\mathbf{p}}$, and $(2 + 2)$-dimensional variables with a double bar like $\bar{\bar{\mathbf{q}}}$.

2   Characteristic Curves of Vector Fields

A curve L is called a *tangent curve* of a vector field $\mathbf{v}(\mathbf{x})$, if for all points $\mathbf{p} \in L$ the tangent vector of L coincides with $\mathbf{v}(\mathbf{p})$.

In a time-dependent vector field $\mathbf{v}(\mathbf{x}, t)$ there are four types of characteristic curves: stream lines, path lines, streak lines, and time lines. We can start a *stream line* in a space-time point $(\mathbf{x}_0, t_0)$, staying in time slice $t = t_0$, by integrating a tangent curve in the vector field $\bar{\mathbf{s}}(\mathbf{x}, t) = (\mathbf{v}(\mathbf{x}, t), 0)^T$. Similarly, *path lines* of the original vector field $\mathbf{v}$ are described as the tangent curves of the vector field $\bar{\mathbf{p}}(\mathbf{x}, t) = (\mathbf{v}(\mathbf{x}, t), 1)^T$ in space-time. Path lines describe the trajectories of massless particles in time-dependent vector fields.

A *streak line* is the collection of all particles set out at different times but the same point location. In an experiment, one can observe these structures by constantly releasing dye into the flow from a fixed position. The resulting streak line consists of all particles which have been at this fixed position sometime in the past. Considering the vector field $\bar{\mathbf{p}}$ introduced above, streak lines can be obtained in the following way: apply a path surface integration in $\bar{\mathbf{p}}$ where the seeding curve is a straight line segment parallel to the t-axis; a streak line is the intersection of this path surface with a hyperplane perpendicular to the t-axis. As shown in [14], streak lines can be described as tangent curves of the vector field

$$\bar{\bar{\mathbf{q}}}(\mathbf{x}, t, \tau) = \begin{pmatrix} \left(\nabla \phi_t^\tau(\mathbf{x})\right)^{-1} \cdot \frac{\partial \phi_t^\tau(\mathbf{x})}{\partial t} + \mathbf{v}(\mathbf{x}, t) \\ 0 \\ -1 \end{pmatrix}, \tag{1}$$

where $\phi_t^\tau(\mathbf{x})$ denotes the flow map computed from particles seeded at $(\mathbf{x}, t)$ and integrated over a time interval τ. We call $\bar{\bar{\mathbf{q}}}$ the *streak line vector field*. It is defined in the domain $D \times T \times \Upsilon$ with $\tau \in \Upsilon$, i.e., $\bar{\bar{\mathbf{q}}}$ is a 4D vector field if the original flow $\mathbf{v}$ is a 2D time-dependent field. The streak lines of a constant time step can be integrated as tangent curves in the subspace $D \times \Upsilon$, i.e., $\bar{\bar{\mathbf{q}}}$ simplifies to

$$\bar{\mathbf{q}}'(\mathbf{x}, \tau) = \begin{pmatrix} \mathbf{w}(\mathbf{x}, \tau) \\ -1 \end{pmatrix}, \qquad t = \text{const}, \tag{2}$$

where $\mathbf{w}(\mathbf{x}, \tau) = \left(\nabla \phi_t^\tau(\mathbf{x})\right)^{-1} \cdot \frac{\partial \phi_t^\tau(\mathbf{x})}{\partial t} + \mathbf{v}(\mathbf{x}, t)$ denotes the spatial components.

A *time line* is the collection of all particles set out at the same time but different locations, i.e., a line which gets advected by the flow. An analogon in the real world is a yarn or wire thrown into a river, which gets transported and deformed by the flow. However, in contrast to the yarn, a time line can get shorter and longer. It can be obtained by applying a path surface integration in $\bar{\mathbf{p}}$ starting at a line with $t = \text{const}$, and intersecting it with a hyperplane perpendicular to the t-axis. Whether time lines can be described as tangent curves of some derived vector field is still an open research question.

See [14] for a more thorough discussion of characteristic curves.
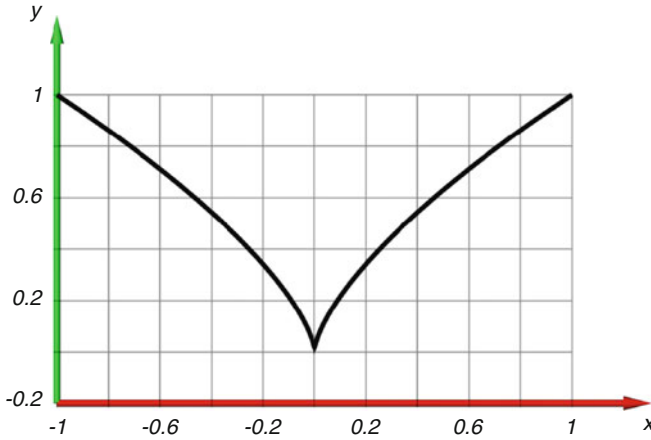
Fig. 1 Plot of the function $x^2 - y^3 = 0$. It has a cusp at the origin, where the tangent vector of the curve abruptly changes its direction

3 Cusps of Characteristic Curves

Cusps are isolated points on a curve where its tangent vector abruptly changes, i.e., the tangent has a discontinuity. Figure 1 shows this for a simple example: the zero-levelset of $x^2 - y^3$, which has a cusp at the origin.

The previously introduced characteristic curves can be described as tangent curves (except for time lines) in their higher-dimensional domains $D \times T$ or $D \times T \times \Upsilon$. Given that the respective tangent curve vector fields $\bar{\mathbf{s}}, \bar{\mathbf{p}}, \bar{\bar{\mathbf{q}}}$ are continuous, their tangent curves are smooth, i.e., they *cannot* have cusps in these higher-dimensional spaces. This follows since the vector field gives the first derivative of the tangent curve: if the vector field is continuous (C^0), then the first derivative of the tangent curve is C^0, which makes the tangent curve itself smooth (C^1).

However, a projection of a characteristic curve into a subspace may very well have a cusp. In this paper, we are interested in a *spatial* projection, since we wish to create static, two-dimensional visualizations of these curves in Sect. 5. We strive for high expressiveness of these visualizations by reducing the amount of clutter that is likely to be introduced by the projection. To that end, we analyze the occurrence of cusps in these projections, since the abrupt turning of the tangent direction at cusps communicates a non-smoothness to the viewer despite the fact that the underlying field is actually smooth or at least continuous. Hence, we want to exclude cusps (and nearby areas) from these visualizations. In the following, we study cusps in spatial projections of all four types of characteristic curves.

Stream lines: The case for stream lines is rather trivial. Stream lines live in a constant time step, i.e., a spatial projection does not alter their geometry and they cannot have cusps in a continuous vector field.
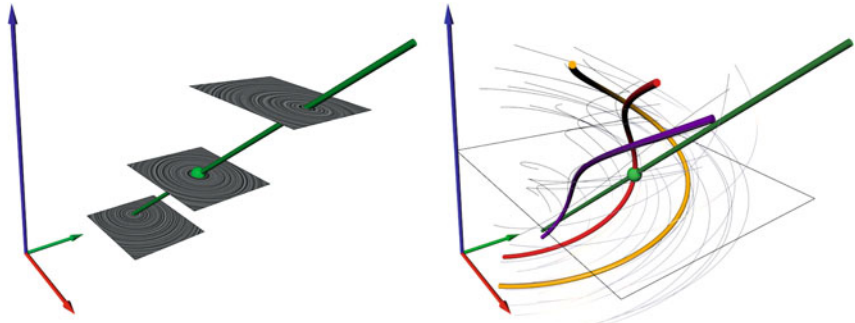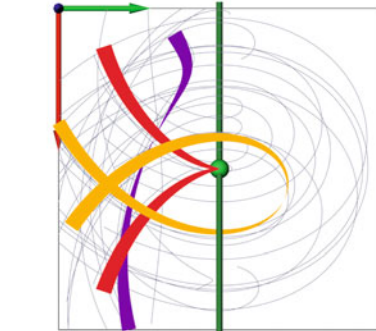
(a) Stream lines shown with LIC. Tracked critical point shown as straight green line.

(b) 28 path lines, three of them highlighted. See below for their spatial projection.
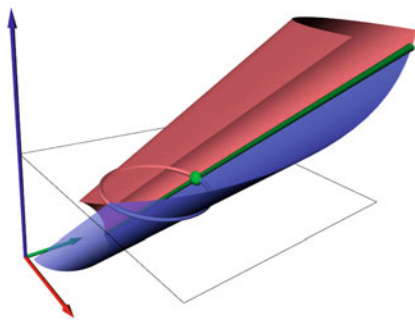
(c) Forward (red) and backward (blue) path surface started from the tracked critical point. Intersection with t_0 yields cusp seeding line.

(d) Spatial projection of the path lines from (b). Highlighted path lines are scaled by distance to t_0 (shown in (b) by the large quad).
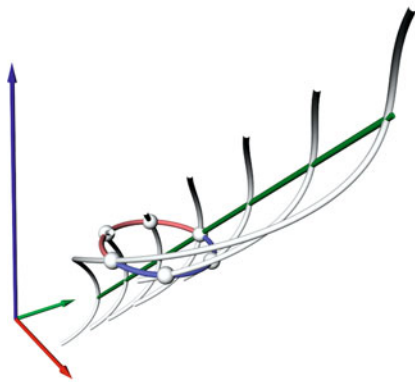
(e) Every path line started on the cusp seeding line (red/blue) runs through the critical point.

(f) Spatial projection of (e). Path lines shown in black and scaled by distance to cusp.

Fig. 2 Cusps in the spatial projection of path lines shown for the 2D time-dependent vector field (3). The *red* and *green* axes denote the spatial (x, y)-domain, whereas the temporal dimension is depicted by the *blue* axis. The time step t_0 is denoted by the large quad in (b) and (c)

Time lines: Although a tangent curve description for time lines is not yet available, we can already remark the following: a time line can be seen as the front of a path surface integration in $\bar{\mathbf{p}}$. Assuming that $\bar{\mathbf{p}}$ is smooth and remembering that it does not have any critical points, the front line undergoes only smooth transformations during the integration. Since t is constant for a time line, a spatial projection will not alter the geometry of the curve. Hence, time lines cannot have cusps. They may however be subject to strong bending during the integration, which may distort the region around some point in a way that resembles a cusp, but those points would not be cusps in the infinitesimal sense.

3.1 Cusps in the Spatial Projection of Path Lines

Path lines of $\mathbf{v}(\mathbf{x}, t)$ are the tangent curves of $\bar{\mathbf{p}}$ in space-time. The first two components of $\bar{\mathbf{p}}$ become zero at locations $\mathbf{c}$ where $\mathbf{v}$ has a critical point, i.e., $\bar{\mathbf{p}}(\mathbf{c}) = (0, 0, 1)^T$. A path line running through $\mathbf{c}$ advances there only in temporal direction, but stays at the same spatial location (for an infinitesimally small time). Consequently, a following spatial projection will lead to a cusp at this point.

We can see this in Fig. 2, where the vector field

$$\mathbf{v}(x, y, t) = \begin{pmatrix} -y \\ x - t \end{pmatrix} \tag{3}$$

is visualized. It contains a critical point of type *center* that moves over time in x-direction. Figure 2a shows its evolution as a straight green line in space-time together with LIC visualizations of the stream lines in three selected time steps. Note that the spatial dimensions in all following visualizations are depicted by the red and green axes, whereas time is denoted by the blue axis.

Figures 2b and d show a number of path lines in space-time and in the spatial projection, respectively. Three of them have been highlighted in pink, red, and yellow. We make the following observations for the spatial projection (Fig. 2d):

- A lot of visual clutter is apparent due to path lines intersecting each other.
- Path lines may have self-intersections as exemplified by the yellow curve.
- Path lines have cusps when they run through a critical point of $\mathbf{v}$ during their integration, as exemplified by the red curve. Note that this curve is perfectly smooth in space-time (Fig. 2b).

We will deal with the (self-)intersection issues in Sects. 4 and 5. Here it remains to understand which subset of all space-time seeding locations gives rise to path lines with cusps in their spatial projection. Later, we will exclude those areas from our seeding algorithm.

The critical points of $\mathbf{v}$ are line-type structures in space-time. They can be extracted with a number of methods, for example using Feature Flow Fields [10,11]. Since all path lines with cusps have to intersect these critical lines, we can collect

all their possible seeding locations using two path surface integrations in $\bar{\mathbf{p}}$ started from each critical line: one in forward and one in backward direction. Figure 2c illustrates this. Intersecting the surfaces with a time step t_0 yields *cusp seeding lines*: they describe all seeding locations at t_0 which give rise to path lines with cusps in their spatial projection. Note that depending on where we start a path line integration on a cusp seeding line, the actual crossing of the critical point might be before, after or at t_0. Figure 2e shows this in space-time, Fig. 2f in space.

3.2 Cusps in the Spatial Projection of Streak Lines

As described in Sect. 2, streak lines live in a constant time step and are given there by (2) as the tangent curves of $\bar{\mathbf{q}}'$ in the 3D domain $D \times \Upsilon$, where Υ refers to the τ-dimension denoting the integration interval. Hence, a projection that removes the τ-dimension is required regardless of whether we visualize streak lines in space-time or just space.

Such a projection leads to cusps, in a similar way as for path lines, at locations $\mathbf{c}$ where $\mathbf{w}$ has a critical point, i.e., $\bar{\mathbf{q}}'(\mathbf{c}) = (0, 0, -1)^T$. It can be shown that this implies $\mathbf{v}(\phi_t^\tau(\mathbf{x})) = \mathbf{0}$. In other words, the critical points of the original flow are closely related (via the flow map ϕ) to the critical points of the spatial components of the streak line vector field.

Figure 3 shows this for the example vector field (3), that we already used in the previous section. Its streak line vector field is given as $\bar{\bar{\mathbf{q}}}(x, y, t, \tau) = (\cos(\tau) - 1 - y, -\sin(\tau) + x - t, 0, -1)^T$. We show only the (x, y, τ)-subspace at $t = 0$, where streak lines are given as tangent curves of $\bar{\mathbf{q}}'(x, y, \tau) = (\cos(\tau) - 1 - y, -\sin(\tau) + x, -1)^T$.
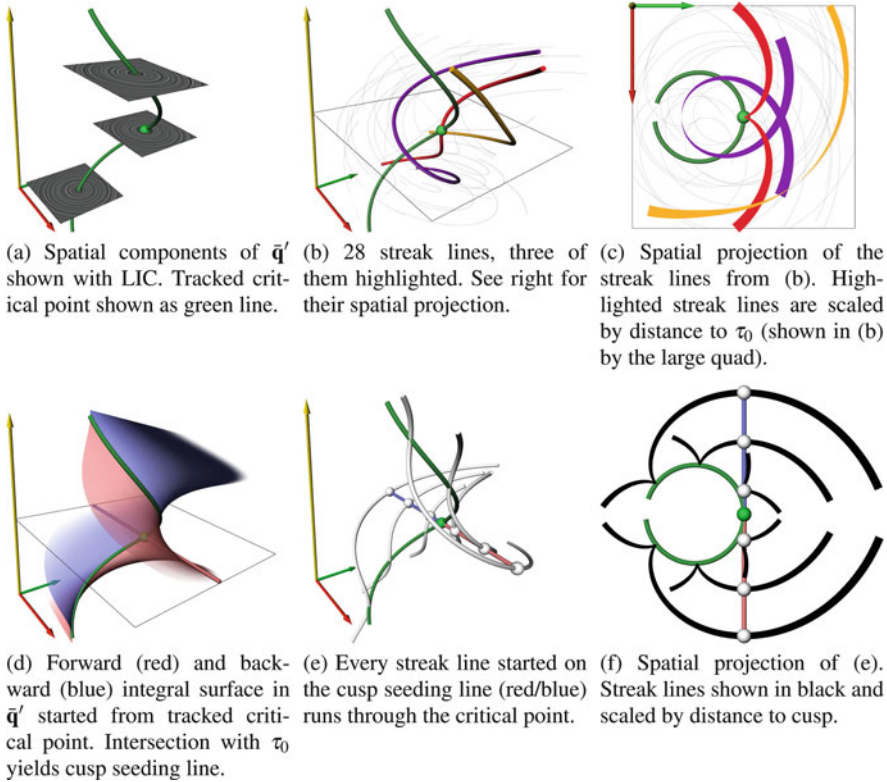
The spatial components of $\bar{\mathbf{q}}'$ are visualized using LIC in Fig. 3a together with the tracked critical point. Note the difference to the tracked critical point of $\mathbf{v}$ shown in Fig. 2a.

Figures 3b and c show a number of streak lines in $D \times \Upsilon$ and in the spatial projection, respectively. The three highlighted streak lines exemplify, that we have to deal with (self-)intersections and cusps for streak lines as well.

We collect all possible seeding locations for streak lines with cusps similar to the path line case: integrate two surfaces (forward/backward) in $\bar{\mathbf{q}}'$ starting from every critical line (Fig. 3d). Their intersection with a certain τ_0 gives the cusp seeding lines, that describe all seeding locations at τ_0 which give rise to streak lines with cusps in their spatial projection (Figs. 3e and f).

## 4   Remarks on Self-Intersections of Characteristic Curves

For the sake of simplicity, the following remarks are made for path lines. They extend to streak lines in a similar manner.

(a) Spatial components of $\bar{\mathbf{q}}'$ shown with LIC. Tracked critical point shown as green line.

(b) 28 streak lines, three of them highlighted. See right for their spatial projection.

(c) Spatial projection of the streak lines from (b). Highlighted streak lines are scaled by distance to τ_0 (shown in (b) by the large quad).

(d) Forward (red) and backward (blue) integral surface in $\bar{\mathbf{q}}'$ started from tracked critical point. Intersection with τ_0 yields cusp seeding line.

(e) Every streak line started on the cusp seeding line (red/blue) runs through the critical point.

(f) Spatial projection of (e). Streak lines shown in black and scaled by distance to cusp.

Fig. 3 Cusps in the spatial projection of streak lines shown for the 2D time-dependent vector field (3). Shown is the (x, y, τ)-subspace (*red, green, yellow axes*) at $t = 0$, where streak lines are given as tangent curves of $\bar{\mathbf{q}}'$. The large quad in (b) and (c) denotes $\tau_0 = 0$

Consider a path line seeded at $(\mathbf{x}, t_0)$ that exhibits a self-intersection in its spatial projection – as it is shown in the leftmost image of Fig. 4. Assume that it is possible to change its spatial seeding location (staying in t_0) such that the loop created by the intersection becomes gradually smaller. Eventually, the loop will degenerate to a point: a cusp. The seeding location is now on a cusp seeding line ℓ_c and the previous seeding locations have been on one side of ℓ_c (in a local sense). Placing the seed on the other side of ℓ_c yields a path line without a self-intersection.

It is easy to see that cusp seeding lines play a vital role in the binary segmentation of a time step into areas where seeding a path line leads to self-intersection(s) and where it does not. Figure 4 illustrates this for our example vector field (3) from the previous section. Here, we have a single cusp seeding line in the shape of a circle. In general, cusp seeding lines are not closed and are of arbitrary shape. It turns out that all seeding locations outside the circle give rise to self-intersecting path lines, whereas only path lines seeded inside the circle do not have a self-intersection in their spatial projection.
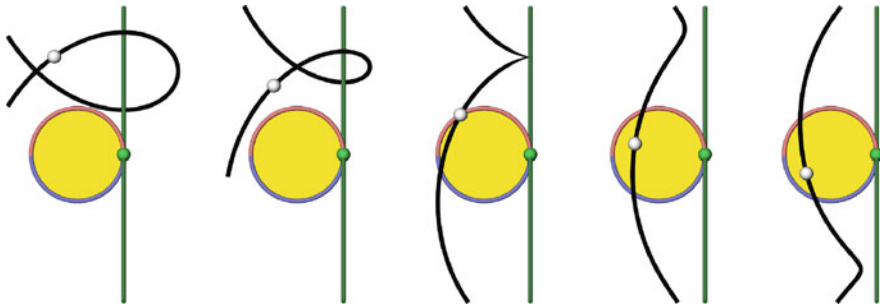
Fig. 4 Gradually changing the seeding location (*gray ball*) of a path line (*black*) leads to a qualitatively different behavior when crossing a cusp seeding line (*red/blue curve*). Shown is the example vector field (3), where only seeding locations inside the circle (*yellow area*) give rise to non-self-intersecting path lines
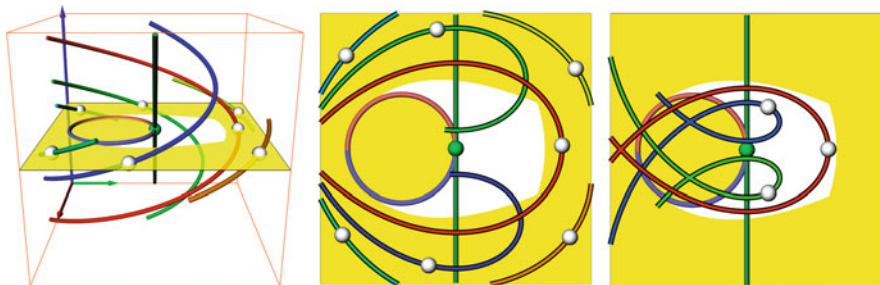
(a) Path lines seeded where the boundary prematurely prevents their self-intersection.

(b) Spatial projection of (a).

(c) Only a rather small part of the domain is left to give rise to self-intersecting path lines.

Fig. 5 Defining a boundary $[-2.5, 2.5]^3$ for the example vector field (3) limits the area (*white*) where seeding path lines leads to their self-intersection in the spatial projection

At least one other component contributes to the self-intersection segmentation of a time step: the boundary of the domain – may this be a boundary given as a hard constraint by a numerical simulation, or as a soft constraint defined by a user. A possible self-intersection of a path line is prevented if the integration stops at the boundary before having reached the intersection point. In other words, introducing a boundary *reduces* the number of self-intersecting path lines. Figure 5 illustrates this.

There is an equivalent to cusp seeding lines associated with the boundary: one could call them "boundary touching lines." They consist of seeding locations at the boundary giving rise to path lines, which have one end touching another part of the spatially projected path line. To one side of a "boundary touching line" we find self-intersecting path lines, to the other side the ones that could not make it to the self-intersection due to the premature integration stop at the boundary.

As a last remark, we also suspect topological separatrices (emanating from saddle points) to play a role in the self-intersection segmentation.

However, for the purposes of this paper, it suffices to understand that only a subset of all possible seeding locations (in some flows actually a rather small subset) gives rise to self-intersecting path lines. This will guide some algorithmic decisions in the following section.

5   Intersection-Aware Visualization of Path and Streak Lines

Our method for creating uncluttered, static visualizations of path or streak lines follows these principles:

- Only a very limited, predetermined number of field lines is allowed to have self-intersections and intersect each other. This is required to highlight turbulent areas.
- All other field lines shall not have any intersections.
- Long field lines are preferred.
- A good coverage of the domain is desired.
- Field lines shall have a certain minimal distance to each other.
- Cusps and cusp-like shapes shall be avoided.

Algorithm 1: Intersection-Free Placement of Field Lines   *(Python-like syntax)*

```python
def FieldLinePlacement(nDesiredLines):
    ResultLines = []
    Pool = IntegrateFieldLines(m) #Initialize with m randomly seeded field lines. default:
      m = 100

    while len(ResultLines) < nDesiredLines:
        Pool += IntegrateFieldLines(n) #Get n new randomly seeded field lines. default: n = 30
        Pool.SortByLength() #Sort field lines in descending order; longest comes first.
        LengthOfLastLongLine = Pool[0].Length()
        ResultLines += Pool.pop(0) #Add the currently longest line to the result.

        while len(ResultLines) < nDesiredLines:
            Pool.CutFieldLinesIntersectingWith(ResultLines) #Intersection-free!
            Pool.SortByLength() #Sort field lines in ascending order; longest comes first.
            if Pool[0].Length() / LengthOfLastLongLine > x: #default: x = 0.98
                ResultLines += Pool.pop(0) #Add the currently longest line to the result.
            else:
                break #Available field lines got too small. Get new ones.

        #Delete small lines from the pool before we add new ones in the next loop.
        Pool.RemoveLinesSmallerThan(LengthOfLastLongLine * y) #default: y = 0.5

    return ResultLines
```

The main ingredient of our method is Algorithm 1, which achieves a completely intersection-free placement of field lines with a reasonable domain coverage and preference for long lines. The basic idea is to randomly seed a large number of field lines, put them in a pool of available lines, and iteratively copy the longest one into the result. After adding a new line to the result, all remaining lines in the pool have to be shortened such that they do not intersect one of the result lines. We continue with the process of adding the currently longest line to the result and shortening the remaining ones in the pool until the lines in the pool become too short. Then we add a number of new field lines to the pool and continue to do so until we have reached a desired number of field lines in the result.

We use a texture-based approach to check for intersections: all field lines in the result are also rendered into a 2D texture. Testing a field line from the pool for intersection with the result lines is then just a matter of checking for overlapping pixels. The texture also allows us to maintain a certain minimal distance between lines by using a certain pixel width when rendering new lines into the texture. This works nicely together with the intersection test and also with the seeding of new field lines, where spatial locations with covered pixels are excluded from the seeding.

Furthermore, we use a simple voxel bit mask to exclude certain space-time locations from the seeding of new field lines. This includes obstacles in the flow (such as a cylinder), and most importantly areas around cusp seeding lines: cusps communicate discontinuities in the flow which are actually not there, since cusps are due to the spatial projection and not due to the underlying flow. We exclude these areas as follows (see also Sect. 3):

- Track all critical points, which yields lines in space-time.
- Integrate two surfaces (forward/backward) from each of these lines.
- Render the surfaces into the voxel bit mask with a certain width such that larger areas around the actual cusp seeding lines are avoided.

Applying this algorithm yields cusp-free and intersection-free visualizations, but it turns out that turbulent parts of the domain are only sparsely or not at all covered since the field lines have numerous (self-)intersections there. We think that an expressive visualization of unsteady flows demands the inclusion of a small number of (self-)intersecting lines as a trade-off between visual clarity and a faithful representation of the flow. Hence, we let the user select a desired number of such lines from an automatically computed set optimized by length and domain coverage, and feed this information into the above algorithm such that the rest of the domain can be filled with non-intersecting field lines. In all visualizations of the following results section we render the selected self-intersecting lines with color (blue for path lines, red for streak lines) and halos to enhance their perception. Furthermore, we encode time into the width of the path lines, and τ into the width of the streak lines.

Streak lines are always seeded at a certain time step, but we allow the user to choose either a time step or a time interval for seeding path lines. A small time interval is suggested as it allows for a better temporal coherence between the path

Fig. 6 Path lines in the flow behind a cylinder have been seeded using Algorithm 1, but without excluding the areas around cusp seeding lines. While the result is free of (self-)intersections, some path lines exhibit cusps or cusp-like shapes (highlighted by *red circles*)

lines. However, due to the boundary and the cutting of the path lines in Algorithm 1, not all path lines will start or end in the same time steps.

Discussion of possible alternatives: Topological information might not be readily available in every visualization system. Simply applying Algorithm 1 without excluding the areas around cusp seeding lines, however, does not yield satisfying results as exemplified in Fig. 6: cusps and cusp-like shapes are clearly visible. Nevertheless, an alternative to our topology-based approach can be formulated based on our theoretical findings from Sects. 3 and 4: we know that cusps appear next to an area of self-intersecting path lines. To exclude them from the seeding, one could determine for every grid point whether it gives rise to a self-intersecting path line. If so, this grid point and a certain number of grid points in its neighborhood are excluded from the seeding. This would serve as the voxel bit mask described earlier, but without actually using topological information. We tested this and it gave results similar to the ones shown in the next section. However, this brute-force method needs more computation time (depending on the resolution of the voxel bit mask) than the topological approach.

6 Results

The following results have been computed single-threaded on a laptop with an Intel Core 2 Duo T9550 (2.66 GHz). The timings for the topological analysis and the subsequent field line placement are given in the respective figure captions. All integrations have been done with a fourth order Runge–Kutta scheme and adaptive step size control.

Figure 7 shows the flow above a heated cylinder which has been simulated using The Gerris Flow Solver [7]. It is given on a $41 \times 70 \times 241$ uniform grid. The comparison between the particle animation and our path line placement clearly elucidates the differences between showing transient impressions and a static visualization comprising all time steps. Furthermore, the seeding of the particles turned out to be a cumbersome, time-consuming process since seeding location and density had to be properly adjusted to achieve the shown effect. With the new path line placement method we were able to produce a meaningful result within seconds.

Path and streak lines for a flow behind a cylinder [7] are shown in Fig. 8 ($400 \times 50 \times 1001$ uniform grid). As reasoned earlier, one finds self-intersecting field lines near critical points of the underlying tangent curve vector field. The topology of
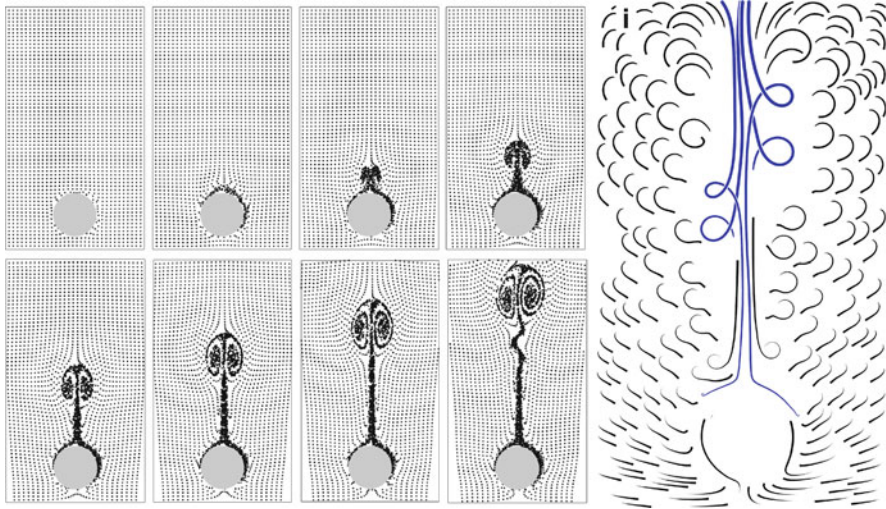
Fig. 7 Heated cylinder data set. Shown is a particle animation (*left*) and the result of our path line placement method (*right*). *Topology/Placement: 6/2 s*
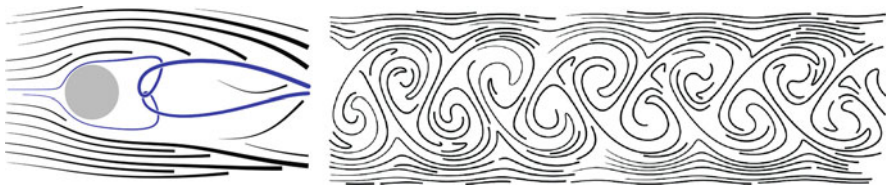
Fig. 8 Flow behind a cylinder. The path lines (*left*) are shown in the vicinity of the cylinder where critical points appear. *Topology/Placement: 2/1 s* The streak lines (*right*) are shown further downstream where the von Krmn vortex street is fully developed. Our algorithm was able to fill the domain with streak lines properly, i.e., no additional streak lines have been seeded. *Topology/Placement: 7/6 s*

the original flow field has only a few critical points directly behind the cylinder, which makes the path lines interesting there while they become rather simple further downstream. The situation is different for streak lines: the spatial components of $\bar{\bar{\mathbf{q}}}$ have critical points where $\mathbf{v}(\phi_t^\tau(\mathbf{x})) = \mathbf{0}$. In other words, the critical points of the original flow have been "transported" downstream and create interesting streak line patterns there.

Figure 9 shows the streak lines of the well-known *Double Gyre* flow. We have chosen a rather long τ-interval for computing the streak line vector field ($256 \times 128 \times 1280$ uniform grid), which yields long streak lines in the final visualization. Two streak lines in the left part had to be selected by the user since this part contains only self-intersecting streak lines.
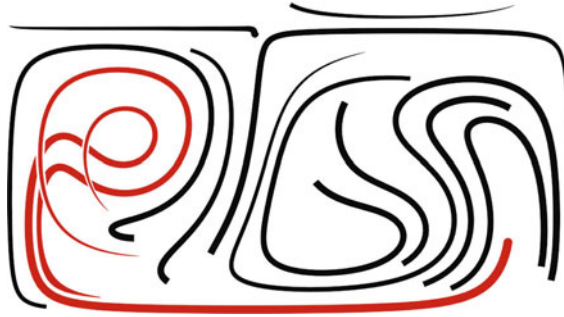
Fig. 9 Streak lines of the Double Gyre flow for $t = 10$ in the τ-interval $[-10, 0]$. The two *red curves* have been selected by the user from a set of pre-computed self-intersecting streak lines in order to fill the domain. Otherwise the left part would not have been filled by Algorithm 1 since it contains mainly self-intersecting streak lines. *Topology/Placement: 0.5/0.5 s*

7 Conclusions and Future Work

We analyzed the spatial projections of characteristic curves in terms of cusps and self-intersections. These insights allowed us to develop a novel placement method for streak and path lines.

Future work needs to address the issue of intersecting field lines. Here, we avoided them as much as possible, but this may also leave certain areas less covered. Also, such visualizations could communicate other properties of the flow (curvature, vorticity, converging/diverging, etc.) by allowing a varying density of the shown field lines. Of course, incorporating intersections and dense areas into an image calls for new rendering approaches for these kinds of visualizations to maintain a certain visual clarity.

References

1. Cuntz, N., Pritzkau, A., Kolb, A.: Time-adaptive lines for the interactive visualization of unsteady flow data sets. Comput. Graph. Forum **28**(8), 2165–2175 (2009)
2. Garth, C., Li, G.S., Tricoche, X., Hansen, C.D., Hagen, H.: Visualization of coherent structures in transient 2D flows. In: Hege, H.C., Polthier, K., Scheuermann, G. (eds.) Topology-Based Methods in Visualization II, Mathematics and Visualization, pp. 1–13. Springer, Heidelberg (2009). Topo-In-Vis 2007, Grimma, Germany, March 4–6
3. Jobard, B., Lefer, W.: Creating evenly-spaced streamlines of arbitrary density. In: Proceedings of 8th Eurographics Workshop on Visualization in Scientific Computing, Boulogne, pp. 57–66 (1997)
4. Kenwright, D.N., Lane, D.A.: Interactive time-dependent particle tracing using tetrahedral decomposition. IEEE Trans. Visual. Comput. Graph. **2**(2), 120–129 (1996)
5. Marchesin, S., Chen, C.K., Ho, C., Ma, K.L.: View-dependent streamlines for 3D vector fields. IEEE Trans. Visual. Comput. Graph. (Proceedings Visualization 2010) **16**(6), 1578–1586 (2010)

6. Mebarki, A., Alliez, P., Devillers, O.: Farthest point seeding for efficient placement of streamlines. In: Proceedings of IEEE Visualization, Minneapolis, USA, 2005, pp. 479–486 (2005)
7. Popinet, S.: Free computational fluid dynamics. ClusterWorld **2**(6), 2–8 (2004). URL http://gfs.sf.net/
8. Rosanwo, O., Petz, C., Prohaska, S., Hotz, I., Hege, H.C.: Dual streamline seeding. In: Proceedings of IEEE Pacific Visualization, Beijing, China, pp. 9–16 (2009)
9. Sanna, A., Montrucchio, B., Arinaz, R.: Visualizing unsteady flows by adaptive streaklines. In: Proceedings of WSCG'2000, Plzen, Czech Republic (2000)
10. Theisel, H., Seidel, H.P.: Feature flow fields. In: Data Visualization 2003. Proceedings of VisSym 03, Grenoble, France, pp. 141–148 (2003)
11. Theisel, H., Weinkauf, T., Hege, H.C., Seidel, H.P.: Topological methods for 2D time-dependent vector fields based on stream lines and path lines. IEEE Trans. Visual. Comput. Graph. **11**(4), 383–394 (2005)
12. Tricoche, X., Wischgoll, T., Scheuermann, G., Hagen, H.: Topology tracking for the visualization of time-dependent two-dimensional flows. Comput. Graph. **26**, 249–257 (2002)
13. Turk, G., Banks, D.: Image-guided streamline placement. In: Proceedings of Siggraph '96, New Orleans, USA, pp. 453–460 (1996)
14. Weinkauf, T., Theisel, H.: Streak lines as tangent curves of a derived vector field. IEEE Trans. Visual. Comput. Graph. (Proceedings Visualization 2010) **16**(6), 1225–1234 (2010). URL http://tinoweinkauf.net/. Received the Vis 2010 Best Paper Award
15. Weiskopf, D.: Dye advection without the blur: a level-set approach for texture-based visualization of unsteady flow. Comput. Graph. Forum (Eurographics 2004) **23**(3), 479–488 (2004)
16. Wiebel, A., Scheuermann, G.: Eyelet particle tracing—steady visualization of unsteady flow. In: Proceedings of IEEE Visualization, Minneapolis, USA, 2005, pp. 607–614 (2005)
17. van Wijk, J.J.: Image based flow visualization. In: Proceedings of ACM SIGGRAPH '02, San Antonio, USA, pp. 745–754 (2002)

Glyphs for Non-Linear Vector Field Singularities

Alexander Wiebel, Stefan Koch, and Gerik Scheuermann

1 Introduction

Vector field singularities (also called critical points), i.e. zeros in the vector field, are a central ingredient to the analysis of vector field topology [10]. In two-dimensional as well as three-dimensional vector fields they are the locations at which separatrices (lines in 2D, surfaces in 3D) originate or end. These separatrices partition the field into areas with streamlines that have the same origin and destination (α and ω limit sets) and thus allow to divide the fields into areas with similar behavior. Unfortunately, this partition is only useful for time-independent vector fields or instantaneous snapshots of time-dependent vector fields. When considering complete time-dependent vector fields the critical points lose their importance concerning a partition of the space. Still, for time-dependent fields, they are important in the development of certain flow features. The usefulness of their tracking and visualization has for example been shown in the context of vortex breakdown [7].

The obvious importance of singularities for the overall flow behavior results in a need for the analysis of the flow in the vicinity of such singularities. In the past vector field singularities themselves have been visualized mainly by showing either only their location with a dot or a sphere, or with glyphs illustrating the nature of their linear approximation. A prominent example of the latter are the iconic representations of the repelling/attracting and rotating nature of the field around the singularities that can be found in a paper by Theisel et al. [21]. They go beyond the

A. Wiebel (✉)
Zuse-Institut Berlin, Takustr. 7, 14195 Berlin-Dahlem, Deutschland
e-mail: wiebel@zib.de

S. Koch · G. Scheuermann
Institut für Informatik, Universität Leipzig, PF 100920, 04009 Leipzig, Deutschland
e-mail: stefan.koch@informatik.uni-leipzig.de; scheuermann@informatik.uni-leipzig.de
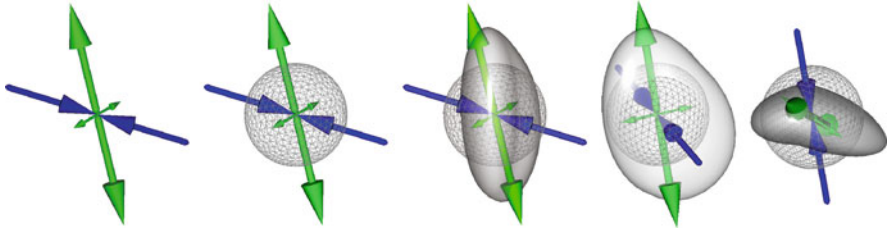
Fig. 1 From *left* to *right*: Saddle point, saddle Point with seeding sphere, saddle point with seeding sphere and deformed sphere illustrating the flow behavior, other perspective of previous image and another perspective of the same image. Note how the glyph (deformed sphere) illustrates the non-linear behavior around the saddle point; linear behavior would produce an ellipsoid (see Sect. 3.2)

common visualization showing only arrows for the directions of the eigenvectors of the linear approximation (Jacobian matrix).

In this paper we want to go even further and illustrate also the non-linear behavior of the field around a singularity. Therefore we introduce glyphs (see Fig. 1) whose shape and coloring provide information about the behavior of the flow gathered during a short time integration in the field. In particular, the main contributions of this paper are:

- An improved and robust computation of the *linear neighborhood* that was introduced to visualization by Schneider et al. [19].
- A new type of glyphs that is seeded according to this *linear neighborhood* and that describes the non-linear behavior of vector field singularities not only in the directions of the eigen-manifolds.
- A demonstration of different additions to the glyphs (coloring, FTLE and streamlets) that emphasize special characteristics of the depicted flow.

As we will show throughout this paper, the introduced techniques can help to improve visualization, understanding and analysis of vector field singularities and their surrounding flow. Please note that we do not aim at specific new findings in application areas. Our method is simply intended to provide additional and complementary views on the vicinity of the singularities. We explicitly recommend a combination with other methods (e.g. [15, 16]) described in Sect. 2.

2 Related Work

Glyphs are very common in the visualization of vector fields. One of the first techniques uses arrows to depict the vectors themselves. It is not attributable a specific author. The same holds for the use of tripods to depict the linear behavior of critical points. A more advanced glyph shown by Theisel et al. [21] was already mentioned in the introduction. Weinkauf et al. [22] presented an icon for higher

order vector field singularities. De Leeuw and van Wijk [3] combine a collection of iconic representations for divergence, rotation, shear, curvature, acceleration and the vector itself to form a glyph called *local flow probe*. All properties shown by the local flow probe are derived from the linear approximation of the field. Additionally, the display of the flow direction and speed, would not work for singular points as the field is zero there. Recently, Hentschel et al. [11] successfully demonstrated the use of glyphs in virtual reality to show the deformation of blood cells in a blood flow.

The technique with the most similarities to our work was reported by Löffelmann et al. [15]. They present abstract depictions of the eigen-manifolds of some types of singularities and streamlets on spheres around singularities to illustrate the dynamical system in the vicinity of the points. Löffelmann and Gröller improved this method by adding streamlets around the characteristic curves of the singularities [16]. In contrast to these methods which focus on the eigen-manifolds of the singularities, the glyphs produced by our method give an impression of a singularity's vicinity in all directions without the need of integral lines started on a sphere around the singularity. We consider the integral line alternative (e.g. the sphere tufts [15]) to yield significantly more cluttered images. Another important difference is that our method can illustrate the interaction of close singularities (see e.g. Fig. 7).

While all above methods dealing with singularities require them to be hyperbolic, hyperbolicity is irrelevant for the construction of the presented glyphs.

Recently glyphs have become very popular to illustrate results from diffusion weighted magnetic resonance imaging [4, 12]. In this context a large number of different glyph types that show the diffusion strengths in the different directions as deformed sphere surfaces has been developed [1,5,17]. Our approach is inspired by this type of visualization.

As we advect a surface around a singularity to obtain the glyph, flow volumes [2] by Becker et al. are also related. However, they do not pay special attention to the characteristics of singularities and their algorithm is much more complex than our basic advection because of the volumetric subdivision. For long integration times our method could also need refinement. The sphere can then be interpreted as a time surface and advected as described by Krishnan et al. [14]. Note, however, that long integration is not intended for the glyphs themselves.

3 Mathematical Foundations

3.1 Linear Neighborhood

For our algorithm we need the concept of a linear neighborhood around a critical point as introduced by Schneider et al. [19]. We define the *linear neighborhood* $U_L(\mathbf{x}_c)$ around a critical point $\mathbf{x}_c \in \mathbb{R}^3$ as the region for which a linear approximation of the vector field holds within a certain bound $C_L \in \mathbb{R}, C_L > 0$:

$$U_L(\mathbf{x}_c) = \left\{ \mathbf{y} \in \mathbb{R}^3 \;\middle|\; \frac{\|\mathbf{v}(\mathbf{y}) - J(\mathbf{x}_c) \cdot (\mathbf{y} - \mathbf{x}_c)\|}{\|\mathbf{v}(\mathbf{y})\|} < C_L \right\} \tag{1}$$

where J denotes the Jacobian. We describe the computation of the neighborhood in Sect. 4.

3.2 Note on Ellipsoids in Linear Vector Fields

We note that the application of the advection scheme we use to produce the glyphs leads to the well known ellipsoidal glyphs in linear vector fields with one isolated singularity.[1] Thus, glyphs for such vector fields can directly be derived from their Jacobian matrix defining the whole field. The comparatively costly step of advecting the sphere is not necessary. This holds to some degree also for the field in the linear neighborhood around a singularity. For any sensible choice of the linearity threshold the difference to an ellipsoid will not be visible.

The fact that ellipsoids always transform to ellipsoids in linear vector fields may not be obvious. Therefore we provide a short discussion in the following. We consider the changes of an ellipsoid under an integral line approximation algorithm because this is what we use to advect the vertices of the sphere. In the following, we use the Euler approximation method for simplicity. It is given by $\mathbf{y}_{n+1} = \mathbf{y}_n + h\,\mathbf{v}(\mathbf{y}_n)$ where $\mathbf{y}_0$ is the start position, h the step size and $\mathbf{v}$ the vector field. In a linear vector field this can be written as

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\, A\mathbf{y}_n \tag{2}$$

where A is the matrix defining the linear vector field and its Jacobian matrix at the same time. Consider all $\mathbf{y}_0$ that make up the sphere (which is a special ellipsoid). If we add the positions of an ellipsoid to those of another ellipsoid the resulting positions will again fulfill the quadratic equation defining an ellipsoid, i.e. form an ellipsoid. The same holds for scaling all positions of an ellipsoid. Together this means, that we need to show that all $A\mathbf{y}_n$ lie on an ellipsoid if all $\mathbf{y}_n$ lie on an ellipsoid in order to show that (2) transforms another ellipsoid to an ellipsoid. Proofs for this statement are available in the literature, see e.g. Hyslop [13].

4 Linear Neighborhood Computation

In contrast to Schneider et al. [19] who only need to guarantee that their points are somewhere in the linear neighborhood for some of their operations, we are explicitly interested in the radius of the largest sphere lying completely in the linear

---

[1]Linear vector fields with one isolated singularity are defined by invertible matrices.

neighborhood. Inside this sphere all positions adhere to the linearity condition. We call the radius of this sphere the radius of the linear neighborhood.

As vector fields are still mostly stored as discrete samples at vertices (building up a grid of cells for interpolation), we compute the sphere using a cell-based procedure outlined by Algorithm 1.

Algorithm 1: Computing the radius of the linear neighborhood

input : Position of singularity: singularityPosition
output: Radius of linear neighborhood: radius

cell ← SearchCell(singularityPosition);
positions ← GetPositions(cell);
posFound ← false;
while *not* posFound **do**
    for pos ∈ positions **do**
        if CheckLinearity(pos) **then**
            add GetNeighboringPositions(pos) to newPositions;
            // no duplicates in newPositions
        else
            posFound ← true;
            outsidePositions ← pos;
        end
    end
    positions ← newPositions;
end
pos ← FindPositionWithMinDistToSing(outsidePositions,singPos);
minimalRadiusFound ← false;
while *not* minimalRadiusFound **do**
    radius ← GetMinLinearRadius(singularityPosition,pos);
    minimalRadiusFound ← true;
    for pos ∈ *discretized ball with* radius **do**
        if *not* CheckLinearity(pos) **then**
            minimalRadiusFound ← false;
            break;
        end
    end
end
return radius;

The linearity test CheckLinearity essentially checks whether the condition in (1) holds for a certain position. FindPositionWithMinDistToSing selects the position from the given set which is closest to the singularity. After this call the grid vertex violating the linearity condition which is closest to the singularity has been found. The distance of this vertex to the singularity is an upper bound for the radius of the linear neighborhood. The second part of the algorithm is concerned with refining the radius. GetMinLinearRadius determines the specific point on a ray between the vertex and the singularity which first violates the linearity

condition. Therefore a binary search[2] is performed. The search is confined to the cell which contains the vertex and which is crossed by the ray. Other cells crossed by the ray are guaranteed to lie completely in the linear neighborhood.

The procedure so far would already compute a reasonable approximation of the radius for most cases. To handle also all other cases we check whether the linearity condition holds for all vertices of a discretized sphere with the radius determined so far. If the check fails for the first time we use the current vertex to reiterate the binary search on a ray from this vertex to the singularity. Again, we get an approximation of the radius and check it on a discretized sphere. We repeat this until we have found a radius for which no vertex on the sphere violates the linearity condition. This radius is the radius of the linear neighborhood. The iteration is guaranteed to terminate, because the radius can only decrease in each step, each step uses a separate binary search and there is a natural minimum (i.e. zero).

5   The Glyphs

5.1   Glyph Generation

As a starting point for the glyph computation, we place spheres with a fixed radius around the singularities of the vector field. The centers of the spheres are exactly at the position of the singularities. The spheres are approximated by repeated subdivision of icosahedra, so that we obtain triangular discretizations for all spheres. The vertices of the triangulations of the start spheres serve as starting points for a integral line computation over a certain time interval. The endpoints of these particle integrations are stored in a so called flow map vector field, which means that the endpoint of every integral line is stored at the corresponding vertex. The simplest version of the glyphs, i.e. the spheres advected respectively deformed by the flow surrounding the singularities, can then be obtained by substituting all sphere vertices by the corresponding flow map entries.

Because the choice of the initial sphere radius and the integration time are crucial for easily interpretable and meaningful glyphs, we will now discuss different radius determination approaches.

5.1.1   Choice of Initial Sphere Radii and C_L

The simplest choice is one user defined radius for all glyphs. This gives the user maximum freedom but may be not the optimal solution concerning depiction of non-linearity.

---

[2]A binary search is suitable here because the error in the linearity condition varies monotonically along the ray. This is immediately clear for tetrahedral cells because linear interpolation is used for them. For other cells, i.e. trilinearly interpolated cells, the monotonicity is not given for the whole cell. However, as the line starts from the "non-linear" vertex closest to the singularity and as the extrema in a trilinear cell lie on the boundary, the ray can only cross monotonic regions.

If the user is not sure which initial sphere size is suitable for the flow field, the initial radius of each sphere may be determined semi-automatically. For this purpose we employ the linear neighborhood introduced in Sect. 3.1, i.e. we use the linear neighborhood spheres as start spheres. Only the parameter C_L has to be given by the user. It indicates how non-linear the field has to be at the starting sphere. The resulting radii are meaningful for each glyph separately, because they will reflect exactly the degree of non-linear flow behavior the user aims to illustrate around each singularity. This is guaranteed as the advection then starts where a certain degree of non-linearity starts. According to the considerations in Sect. 3.2 very small C_L will lead to nearly ellipsoidal glyphs. Note, that the semi-automatic choice results in separate starting radii for the different singularities.

5.1.2   Choice of Integration Time

Besides the choice of the radii for the initial spheres, the integration time for the integral line computations is very important to obtain meaningful glyphs. Again, users can either select the value manually or use an automatically computed integration length for each single initial sphere.

We determine the automatic integration time such that the vertices of a starting sphere are advected approximately a distance (arc length) that is equal to half the radius of that sphere. In fact, we determine the median integration time all vertices of the sphere need to travel this prescribed arc-length. This integration is then used to advect all vertices of the sphere to their final positions. Thus the glyphs are between half and 1.5 times the size of the starting sphere.

5.2   Rendering Styles

To elevate the characteristic of the glyphs we provide different color maps and a glyph evolution.

5.2.1   Color Maps

Color mapping on the glyphs can support the perception of the deformation of the initial sphere through the non-linear flow behavior and can be used to add additional information. We provide two color maps for the first and one color map for the latter task.

First, the value range of the radii can be mapped on our glyphs. As a result of this, we obtain depictions of large values at the positions on the glyphs which have a large distance to their singularity and vice versa we depict small values at the positions that are close to the singularity. Therefore, this color map emphasizes the perceptions of the glyph shape and supports its fast and easy interpretation by the user.
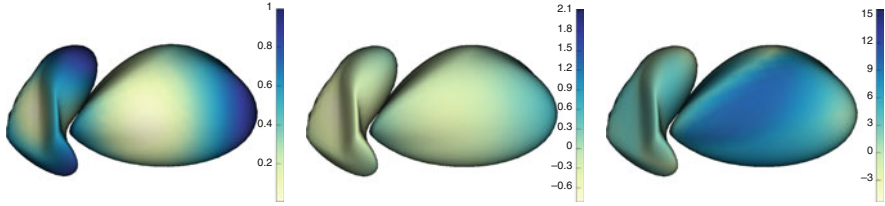
Fig. 2 The images show a selection of color maps on glyphs for two close singularities (from *left* to *right*): normalized distance to singularity, signed distance to start position and FTLE$^+$

The second color map we provide makes a statement about the deformation of the glyph relative to the initial sphere. For that purpose, we subtract the radius at all positions of a glyph from their actual distance to the singularity. Thus, we obtain a positive value if a position moved outside the initial sphere and a negative value if the position moved to the inside. Hence, the user is able to get a quick overview of the glyph shapes relative to the initial spheres without the need to render the spheres.

Coloring glyphs with Lagrangian flow properties depicts divergence and convergence of the flow in the different directions around the singularity. Therefore, we adapt the surface FTLE technique introduced by Garth et al. [8]. In their paper they present FTLE computed for surfaces with a small offset from the boundary of flow embedded surface structures to depict separation of flows. We compute the surface FTLE for our starting spheres. The resulting scalar field on the glyph is mapped to color. An example of this version of our glyphs is shown in Fig. 2 in the right images. Large values indicate a strong stretching of the spherical surface.

Note that the computation of the FTLE causes only a very small amount of extra computation time because the most expensive part, i.e. the advection of the vertices of the starting sphere, is already done for the construction of the glyphs. Only the evaluation of the FTLE values from the flow map has to be performed additionally. For details on the FTLE computation on surfaces we refer the reader to the mentioned paper [8] and for more in depth introduction of FTLE itself to works by Shadden et al. [20] and Haller [9].

5.2.2 Glyph Evolution

Besides the just introduced glyph shapes and color maps, another way to give a deeper insight into the non-linear flow is to illustrate the deformation of the initial sphere by showing the evolution of the glyphs over their integration time.

Therefore, we provide the option to interrupt the glyph integration at equidistant time steps and store the intermediate shapes. Displaying all intermediate shapes with a color coding of the different advection times provides an instructive visualization. An example for this illustration is given in Fig. 3.

Fig. 3 Evolution of a glyph over time. Four steps between initial sphere and final integration time. The *arrows* indicate the linear nature of the singularity for reference. The *right image* is a high resolution version of the glyph in the *left image* with color mapping
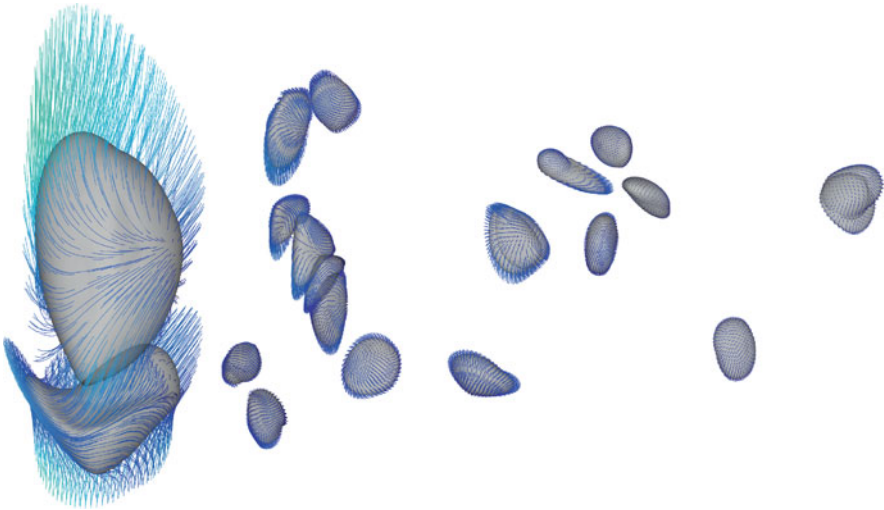
Fig. 4 Glyphs with streamlets emphasizing the flow direction. *Left*: Pair of singularity glyphs. *Right*: Overview of a group of singularities in the gas furnace chamber data set

5.3 Streamlets

In general, our singularity glyphs do not represent rotational flow behavior well. The color map encoding the movement of the different vertices can give a first hint in this regard. However, we generate an additional cue for the flow direction. Therefore we use all glyph vertices as seeding positions for streamlets, i.e. short streamlines (or path lines in the unsteady case). As integration time we choose half the integration as for the glyph surface. The streamlets provide the desired directional cues without obstructing the visibility of the glyph surface. This is demonstrated in Fig. 4.

In contrast to Löffelmann et al. [15] who seed their streamlets stochastically on a sphere, we use the triangulation of the glyph for seeding. This strategy increases the streamlet density where the flow converges and decreases it where the flow diverges, thus providing additional information about the flow behavior.

5.4   Steady vs. Unsteady Flow

Up to now we described the technique without any focus on time-dependent or time-independent vector fields. Fortunately, the whole idea applies to time-dependent as well as time-independent fields without any major changes. The advection of the sphere vertices is done exactly the same way. The only difference is that their traces are path lines instead of streamlines in the unsteady case.

As we discussed earlier, singularities in time-dependent fields have a somewhat different meaning than in steady fields. Thus they deserve some special comments. First, we can simply use instantaneous vector field singularities as in the steady case. However, in the unsteady case there also exist other definitions of meaningful critical points. The most recent definition are the motion compensated critical points given by Fuchs et al. [6]. They are not the obvious singularities in the vector field, but vector field singularities in a specially chosen local frame of reference. As these singularities are also given as simple points, they fit well into our approach and can be used as centers for the start spheres. The advection of the points can then be done either in the originally given field or in the field in the local frame of reference. The choice depends on the desired meaning: the first simply depicts the flow around the critical point the second puts a special emphasis on the critical points influence on its neighborhood.

5.5   Implementation Details

The basis for the glyphs are spheres that have been approximated with the triangular discretization. As described above, the triangulation is derived by repeated subdivision of an icosahedron. Except where mentioned explicitly, all images in this paper use a subdivision depth of three yielding 1,280 triangles for each sphere. This resolution was sufficient to capture all desired details in our experiments, while being coarse enough to be easily handled by any graphics card with simple 3D support.

5.6   Special Cases of Glyphs

Figure 5 shows glyphs for singularities close to the boundary. The radius of the start sphere of some of these glyphs (arrows) is larger than their distance to the boundary. Thus some of their points lie outside the data domain. These points need special treatment because they cannot be advected at all. Two possibilities seem useful. One is discarding all triangles containing such a point. This leaves a hole where the problematic points are. The hole nicely indicates the fact of non-advectable points while producing no visual artifacts. Increasing the resolution of the starting spheres yields smaller holes. This looks nicer but is very expensive regarding time
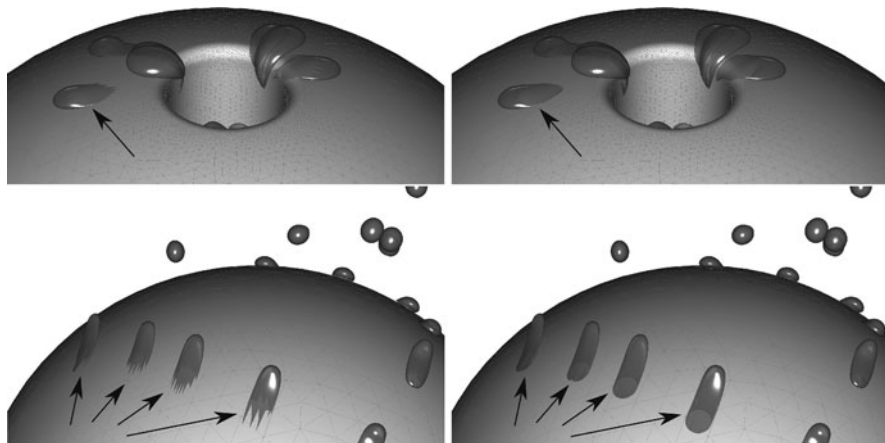
Fig. 5 Singularity glyphs with manually selected radius close to the boundary of a flow embedded ball with a hole. Some glyphs (some marked with *arrows*) are open on one side because the radius was chosen larger than their distance to the wall. The *upper right image* shows glyphs with higher resolution than in the *upper left image*. The *lower right image* shows the same glyphs as in the *lower left image* but with enabled clipping

and memory. The second possibility is to clip the mentioned triangles at the data domain boundary. This results in glyphs nicely attached to the boundary. Jagged parts in theses glyphs indicate shear flow close to the boundary (which is quite common). This is also quite costly in our naive implementation. However, this is only needed for producing final high quality images, and a much more effective implementation is easily achieved by using acceleration data structures like octrees for the intersection step.

6 Application Examples and Discussion

Most of the examples used in this paper are singularities from a vector field representing the flow in a furnace chamber of a central heating system as used in single-family homes. As can be seen in the overview of the dataset given in Fig. 6, the flow contains many (87) isolated singularities as it is very turbulent. The turbulence in this example is not a problem of the design, it is intended to ensure a good intermixture of air and gas for an effective combustion process.

The large number of singularities in this data set leads to many very close singularity pairs. The interaction of such singularities is nicely visible in Fig. 7. The flow around the left singularity (a source) in the left image is repelling and the right singularity (a saddle) attracts the flow in one direction. The expansion of the left glyph causing a "dent" in the right glyph illustrates this effect. A similar illustration is shown in the right image where the attracting and repelling parts of two saddle
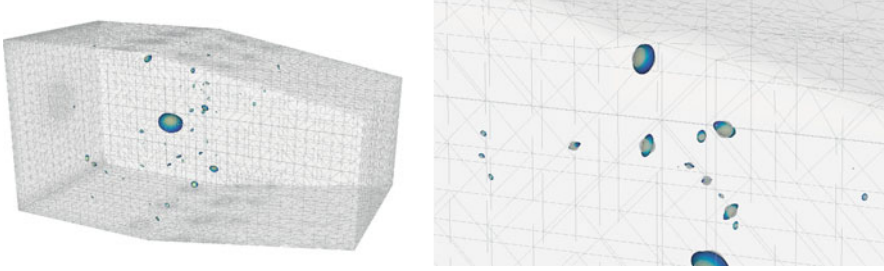
Fig. 6 *Left*: Overview of glyphs for all singularities of the turbulent furnace chamber dataset. *Right*: Zoomed in and slightly rotated version of the same visualization reveals the details. All images are produced with automatically determined radius and integration time
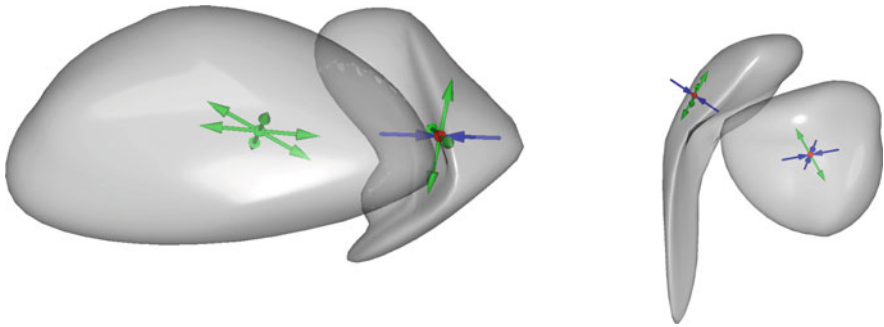
Fig. 7 Pairs of glyphs for close-by singularities with radius larger than linear neighborhood

points interact. Such an interaction is not representable with any other previous technique, like for example the eigenvector arrows shown in the same image.

Figure 5 shows glyphs advected in a time-dependent data set. It represents the flow around a ball with a hole through its center. We illustrate instantaneous vector field singularities taken from a single time step. The special properties of the glyphs in Fig. 5 have been discussed in Sect. 5.6. We compute all information for all rendering styles in one step before displaying the glyphs. This allows us to switch between the different renderings immediately. The computation time of the preprocessing step ranges from 5 to 30 s depending on the chosen glyph parameters for datasets with less than 100 singularities. Our naive special treatment of glyphs close to the boundary (mentioned above) is not included in these numbers. The measurements where performed on one core of an Intel(R) Xeon(R) CPU with 2.40 GHz.

7 Conclusion

We have presented a new type of glyphs for vector field singularities that is capable to illustrate a number of the characteristics of the non-linear behavior of the field around the singularities. Users can determine the size and meaning of the glyphs by either prescribing an initial radius or a linearity threshold. Changing the latter they can explore the different degrees of non-linearity around the singularities. The glyphs are applicable for time-independent as well as time-dependent vector fields. The concept can be easily extended to 2D using circles instead of spheres.

In order to keep the flow behavior and its representation by the glyph shape consistent, some limitations of the overall technique are unavoidable. For example, we can not scale the glyphs to a size that would make it possible to see an overview of the whole field with all glyphs in sufficient detail. If we increase the size of the glyphs after the advection step, the consistence is lost. The glyphs would mislead the observer to believe that the flow in the volume covered by the glyph is represented by the glyph although only the flow in a small part of the volume would be responsible for the shape. Thus, we decided to show the glyphs in their original size (which is large enough to be noticed in an overview, see Fig. 6) and leave it to the user to zoom in to see the details of the local behavior. A future extension simplifying the analysis of the singularities in the context of the whole dataset could be an interface similar to the *interactive closeups* presented in the context of medical exploration by Ropinski et al. [18]. We recommend to show the starting spheres together with the glyphs in general (e.g. Fig. 1). This allows the user to put the deformation in relation to the original shape. In the paper we have omitted the spheres in most cases.

As future enhancement, we plan to depict the unsteadiness as introduced by Fuchs et al. [6] by a color map on the glyphs. We are already working on an extraction of the exact shape of the linear neighborhood (not as sphere).

References

1. Alexander, D.C., Barker, G.J., Arridge, S.R.: Detection and modeling of non–gaussian apparent diffusion coefficient profiles in human brain data. Magn. Reson. Med. **48**(2), 331–340 (2002)
2. Becker, B.G., Lane, D.A., Max, N.L.: Unsteady flow volumes. In: Nielson, G.M., Silver, D. (eds.) Proceedings of the 6th Conference on Visualization '95, pp. 329–335. IEEE Computer Society, Washington (1995)

3. de Leeuw, W.C., van Wijk, J.J.: A probe for local flow field visualization. In: Proceedings of the 4th Conference on Visualization '93, VIS '93, pp. 39–45. IEEE Computer Society, Washington (1993)

4. Domin, M., Langner, S., Hosten, N., Linsen, L.: Direct glyph-based visualization of diffusion mr data using deformed spheres. In: Linsen, L., Hagen, H., Hamann, B. (eds) Visualization in Medicine and Life Sciences, pp. 177–195, 321. Springer-Verlag, New York (2007)

5. Frank, L.R.: Characterization of anisotropy in high angular resolution diffusion-weighted MRI. Magn. Reson. Med. **47**, 1083–1099 (2002)

6. Fuchs, R., Kemmler, J., Schindler, B., Waser, J., Sadlo, F., Hauser, H., Peikert, R.: Toward a lagrangian vector field topology. Comput. Graphics Forum **29**(3), 1163–1172 (2010)

7. Garth, C., Tricoche, X., Scheuermann, G.: Tracking of vector field singularities in unstructured 3D time-dependent datasets. In: Rushmeier, H., Turk, G., van Wijk, J.J. (eds.) Proceedings of the IEEE Visualization 2004 (VIS'04), pp. 329–336. IEEE Computer Society, Washington (October 2004)

8. Garth, C., Wiebel, A., Tricoche, X., Joy, K., Scheuermann, G.: Lagrangian visualization of flow-embedded surface structures. Comput. Graphics Forum **27**(3), 1007–1014 (2008)

9. Haller, G.: Lagrangian structures and the rate of strain in a partition of two-dimensional turbulence. Phys. Fluids **13**(11) (2001)

10. Helman, J.L., Hesselink, L.: Surface representations of two- and three-dimensional fluid flow topology. In: VIS '90: Proceedings of the 1st Conference on Visualization '90, pp. 6–13. IEEE Computer Society Press, Los Alamitos (1990)

11. Hentschel, B., Tedjo, I., Probst, M., Wolter, M., Behr, M., Bischof, C.H., Kuhlen, T.: Interactive blood damage analysis for ventricular assist devices. IEEE TVCG **14**(6), 1515–1522 (2008)

12. Hlawitschka, M., Scheuermann, G.: Hot-lines: Tracking lines in higher order tensor fields. In: IEEE Visualization, p. 4. IEEE Computer Society, Washington (2005)

13. Hyslop, J.: Linear transformations and geometry. Edinburgh Math. Notes **25**, iv–x (1930)

14. Krishnan, H., Garth, C., Joy, K.: Time and Streak Surfaces for Flow Visualization in Large Time-Varying Data Sets. IEEE Transactions on Visualization and Computer Graphics 15(6): pp. 1267–1274 Nov.-Dec. 2009

15. Löffelmann, H., Doleisch, H., Gröller, E.: Visualizing dynamical systems near critical points. In: Proceedings of the Spring Conference on Computer Graphics and its Applications 1998, pp. 175–184. Budmerice, Slovakia (April 1998)

16. Löffelmann, H., Gröller, E.: Enhancing the visualization of characteristic structures in dynamical systems. In: Bartz, D. (ed.) Visualization in Scientific Computing '98, pp. 95–68. Eurographics, Springer-Verlag, New York (1998)

17. Özarslan, E., Mareci, T.H.: Generalized diffusion tensor imaging and analytical relationships between diffusion tensor imaging and high angular resolution diffusion imaging. Magn. Reson. Med. **50**, 955–965 (2003)

18. Ropinski, T., Viola, I., Biermann, M., Hauser, H., Hinrichs, K.: Multimodal Visualization with Interactive Closeups. In: Tang, W., Collomosse, J. P. (eds.), EG UK Theory and Practice of Computer Graphics, Cardiff University, United Kingdom, 2009. Proceedings. Pages 17–24, Eurographics Association, 2009.

19. Schneider, D., Reich, W., Wiebel, A., Scheuermann, G.: Topology aware stream surfaces. Comput Graphics Forum **23**(3), 1153–1161 (2010)

20. Shadden, S.C., Lekien, F., Marsden, J.E.: Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. Physica D **212**, 271–304 (2005)

21. Theisel, H., Weinkauf, T., Hege, H.-C., Seidel, H.-P.: Saddle Connectors – An Approach to Visualizing the Topological Skeleton of Complex 3D Vector Fields. In: Proceedings of IEEE Visualization 03. pp. 225–232, IEEE Computer Society Press, 2003.

22. Weinkauf, T., Theisel, H., Shi, K., Hege, H.-C., Seidel, H.-P.: Extracting higher order critical points and topological simplification of 3D vector fields. In: Proceedings of IEEE Visualization 2005, pp. 559–566. Minneapolis (October 2005)

2D Asymmetric Tensor Field Topology

Zhongzang Lin, Harry Yeh, Robert S. Laramee, and Eugene Zhang

1 Introduction

We define the topology of two-dimensional (2D) asymmetric tensor fields, whose analysis can benefit a wide range of applications in solid and fluid mechanics [23,24]. Topology-based analysis has achieved much success in the processing and visualization of scalar fields [1,6], vector fields [2,8,13,14], and symmetric second-order tensor fields [3,16,21]. Not only can topology provide a concise representation of the fields of interest in these applications, it also enables a framework for systematic multi-scale analysis and temporal feature tracking.

There has been relatively little work in asymmetric (second-order) tensor fields [23,24]. Dodd [5] develops a method to represent the geometry of a symmetric tensor field in terms of its geodesics. The method can be used for a global realization of the tensor field, which allows the user to identify the presence of inconsistencies and singularities in the data. To the best of our knowledge, asymmetric tensor field topology has not been defined, even in the 2D case. Asymmetric tensor fields appear to have richer structures, in terms of both the number and the variety, than vector fields and symmetric tensor fields. We define the topology of a 2D asymmetric tensor field in terms of two topological graphs: *eigenvector graph* and *eigenvalue graph*, based on the eigen-analysis of the tensor field. The nodes of these graphs correspond to regions of tensor field features. The graphs not only describe adjacency relationships between the regions, but also dictate the topological constraints that need to be satisfied by any tensor field simplification

Z. Lin · H. Yeh · E. Zhang (✉)
Oregon State University, Corvallis, OR 97331-8655, USA
e-mail: lin@eecs.oregonstate.edu; harry@engr.orst.edu; zhange@eecs.oregonstate.edu

R.S. Laramee
Swansea University, Wales, UK
e-mail: r.s.laramee@swansea.ac.uk

procedures. We also provide algorithms to construct the graphs given an asymmetric tensor field.

The rest of the chapter is organized as follows: we review related work in Sect. 2, and provide the definition for eigenvalue and eigenvector graphs as well as an algorithm to extract them in Sect. 3. We demonstrate the usefulness of these graphs by applying them to some simulation data in Sect. 4. In Sect. 5, we conclude and discuss some possible future directions.

2   Related Work

Vector field topology: Much work has been devoted to the extraction and visualization of vector field topology [11]. Helman and Hesselink define the topology for vector fields representing fluid flow and propose an analysis and visualization framework [8]. Scheuermann et al. [13] suggest a novel approach to detect higher-order singularities. Several algorithms have been proposed for extracting periodic orbits [2, 19, 20], an important constituent of vector field topology. Tricoche et al. [18] suggest a singularity tracking method for time-dependent vector fields.

To deal with noise in the data, various vector field simplification techniques have been proposed. First-order singularities are either merged into higher-order ones [15] or removed through systematic pair cancellation operations [14, 22]. The simplified topology is easier to perceive and understand while the most prominent structures of the original vector field are maintained.

Symmetric tensor field topology: The topology of symmetric tensor fields is also well studied. Delmarcelle and Hesselink [3] introduce hyperstreamlines for symmetric tensor fields. In addition, they visualize asymmetric tensor fields by using hyperstreamlines for symmetric tensor components while encoding non-symmetric components in an additional vector field.

Delmarcelle and Hesselink [4] and Hesselink et al. [9] define the topology for 2D and 3D symmetric tensor fields. Zheng et al. [25] point out that the degenerate features in 3D tensor fields form curves and propose a fast computation method for integrating degenerate lines in 3D symmetric tensor fields. Furthermore, a number of vector field simplification techniques have been adapted to tensor fields [16, 17, 21].

Asymmetric tensor field analysis: In contrast to symmetric tensor fields, there is relatively little work focusing on asymmetric tensor fields. However, tensor fields that appear in many engineering phenomena and problems are asymmetric in nature, such as the velocity gradient tensors of fluid flow and the deformation gradient tensors in solid medium. Zheng and Pang [24] define degeneracies of asymmetric tensor fields based on singular value decomposition of the tensor field. They also introduce the concept of dual-eigenvectors which allow directional information contained in the tensor field to be visualized even when real-valued eigenvectors do not exist. Zhang et al. [23] introduce the notions of eigenvalue manifold and eigenvector manifold, which are supported by tensor re-parameterizations with physical meaning and lead to effective visualization techniques. To the best of

our knowledge, the topology of asymmetric tensor fields has not been defined nor systematically studied.

3 Asymmetric Tensor Field Topology

In this section, we define asymmetric tensor field topology in terms of two graphs based on eigen-analysis, i.e., the eigenvector graph and the eigenvalue graph.

3.1 Eigenvector Graph

A 2D asymmetric tensor has either two real eigenvalues (real domain) or a pair of complex conjugate eigenvalues (complex domain). Unlike a symmetric tensor, the eigenvectors corresponding to the two real eigenvalues are not orthogonal even in the real domain, and a separate treatment defining the dual-eigenvectors is needed in the complex domain. Zhang et al. [23] define the eigenvector manifold as a sphere illustrated in Fig. 1. The equator of the sphere corresponds to symmetric tensors (pure anisotropic stretching), while the poles represent anti-symmetric tensors (pure rotations). The north and south 45° latitudes are boundaries between the real and complex domains, with the equator inside the real domain and the poles inside the complex domain. These arcs divide the sphere into four regions: $W_{r,n}$, $W_{r,s}$, $W_{c,n}$, and $W_{c,s}$. The subscripts r and c denote the real and complex domains, and n and s represent the northern and southern hemispheres, respectively. A real or complex region in the northern hemisphere has a counterclockwise rotational flow, while a region in the southern hemisphere has a clockwise rotational flow. A tensor field

Fig. 1 Eigenvector manifold: the orientation of the rotational component is counterclockwise in the northern hemisphere and clockwise in the southern hemisphere. Each hemisphere is partitioned into real domains and complex domains. The equator represents pure symmetric tensors (irrotational flows), while the poles represent pure rotations [23]
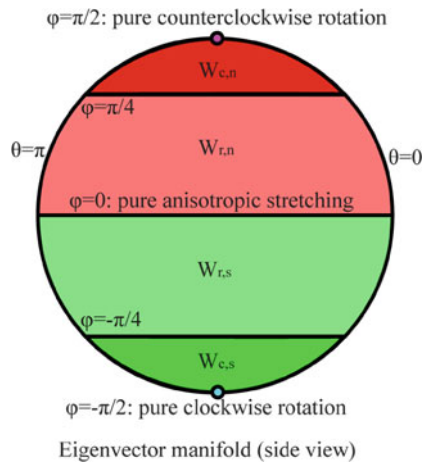
$T(\mathbf{p})$ introduces a continuous map τ_T from the domain of T to the eigenvector manifold, whose inverse $\beta_T = \tau_T^{-1}$ leads to a partition of the domain of T into a collection of four types of regions ($W_{c,n}$, $W_{r,n}$, $W_{r,s}$, and $W_{c,s}$). The boundaries of these regions correspond to the equator and north and south 45° latitudes. The pre-image of the poles are referred as the *degenerate points*.

We define the eigenvector graph of T such that each node in the graph corresponds to a (connected) region in the partition. In addition, every degenerate point is treated as a node. The edges in the eigenvector graph represent the adjacency relationship among the regions (including degenerate points). Due to the continuity of τ_T and β_T, the following adjacency relationships are *not possible*: (1) $W_{r,n}$ and $W_{c,s}$, (2) $W_{r,s}$ and $W_{c,n}$, and (3) $W_{c,n}$ and $W_{c,s}$. For velocity gradient tensors, (3) dictates that it is impossible to transition from a vortex core of counterclockwise rotation ($W_{c,n}$) into a vortex core of clockwise rotation ($W_{c,s}$), or vice versa, without going through the real domain, i.e., stretching-dominated region. Such constraints, when applied to a sequence of slices of a 3D data, have the potential of helping domain experts understand the evolution of flow features over time or in space as well as during continuous multi-scale analysis.

Figure 2 shows two slices from a diesel-engine simulation data set [10]. The four types of regions $W_{c,n}$, $W_{r,n}$, $W_{r,s}$, and $W_{c,s}$ are expressed using light red, dark red, dark green, and light green, respectively. The textures in the background depict the vector field.

3.2 Eigenvalue Graph

Zhang et al. [23] re-parameterize the set of 2×2 asymmetric tensors as follows:

$$\gamma_d \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \gamma_r \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} + \gamma_s \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix} \tag{1}$$

where $\gamma_d = (a + d)/2$, $\gamma_r = (c - b)/2$, and $\gamma_s = (\sqrt{(a - d)^2 + (b + c)^2})/2$ are the strengths of isotropic scaling, rotations, and anisotropic stretching, respectively, and θ provides the directions of the stretching. Based on this parameterization, any non-zero asymmetric tensor can be mapped onto the *eigenvalue manifold* $\{v = (\gamma_d, \gamma_r, \gamma_s) \mid v \cdot v = 1 \text{ and } \gamma_s \geq 0\}$ as shown in Fig. 3. There are five special points in the eigenvalue manifold, corresponding to $v = (1, 0, 0)$ (positive isotropic scaling D^+), $(-1, 0, 0)$ (negative isotropic scaling D^-), $(0, 1, 0)$ (counterclockwise rotation R^+), $(0, -1, 0)$ (clockwise rotation R^-), and $(0, 0, 1)$ (anisotropic stretching S). A tensor T is said to be dominated by one of the above five characteristics, if the image of T onto the eigenvalue manifold has a smaller spherical distance to the special point than any of the other special points. The inverse of this projection leads to a partition of the domain of the asymmetric tensor field, as each region in the partition is dominated by the same characteristic. The region dominated by a
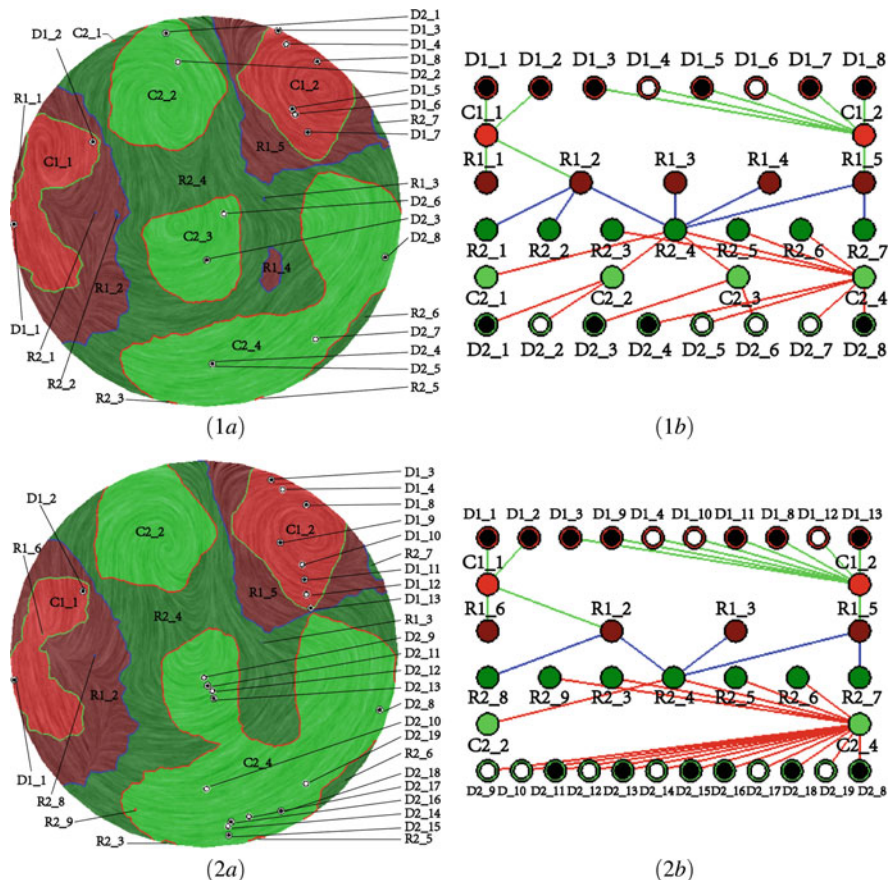
Fig. 2 This figure illustrates the eigenvector graphs applied to two slices of a 3D engine simulation data set [12]. The slices are cut at the 63 mm (1*a*) and 73 mm (2*a*) from the top of the cylinder in the diesel engine simulation [10]. Note that the icons showing nearby degenerate points can overlap during rendering, such as $D2\_4$ and $D2\_5$ in (1*a*). In addition, some regions are too small to be visible without zooming. For example, $R1\_3$ in (2*a*) is a small $W_{r,n}$ region (*dark red*) surrounded by $R2\_4$, a large $W_{r,s}$ region (*dark green*)

characteristic is expressed using colors: D^+ in yellow, D^- in blue, R^+ in red, R^- in green, and S in white (Fig. 4).

We define the eigenvalue graph as follows: the nodes in the graph correspond to the regions in the partition, while each edge encodes the adjacency relationship between a region pair. Notice that two regions may share more than one boundary segment. In this case, each boundary segment will be encoded into an edge in the eigenvalue graph. Figure 4 provides an example illustration using the same data shown in Fig. 2. In Fig. 4(1a), there are two segments between regions $R1\_1$ and S_5. Consequently, there are two edges between them in the graph. Notice that not
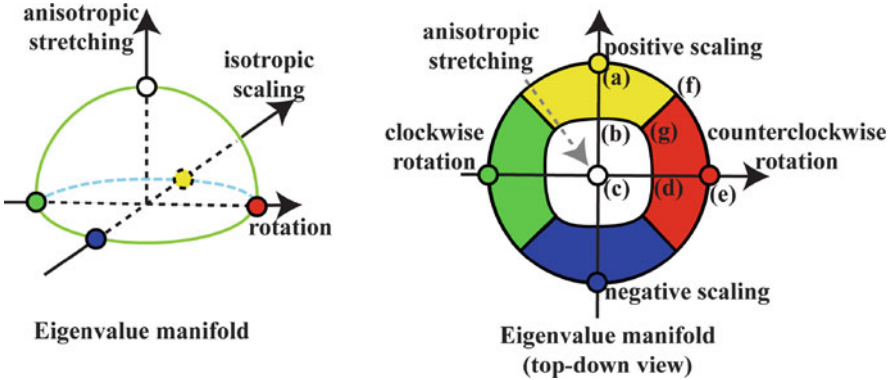
Fig. 3 Eigenvalue manifold: there are five special points on the manifold, which are *positive* and *negative scaling*, *counterclockwise* and *clockwise rotation*, and *anisotropic stretching*. The Voronoi decomposition with respect to these five special points partitions the manifold into five cells, where the flow is dominated by different characteristics [23]

every configuration pair is possible, such as a R^+ and R^- pair. In this case, any curve connecting the two regions must pass through some other type of region. This is a topological constraint that is specific to asymmetric tensor fields. Moreover, the eigenvalue graph can be considered as a 2D cell complex [7], where each 2D cell corresponds to a *junction point* shared by three regions (a tensor there satisfies $|\gamma_d| = |\gamma_r| = \gamma_s$). The cell complex must satisfy that the alternating sum of the number of cells (vertices, edges, and faces) equals the Euler characteristic of the underlying domain. This is yet another aspect of asymmetric tensor field topology.

3.3 Graph Construction Algorithms

We now describe our algorithm to compute the eigenvector and eigenvalue graphs. In our setting the underlying domain is represented by a triangular mesh. The tensor values are defined on the vertices of the mesh and propagated into edges and triangles through the interpolation scheme described in [14].

The most challenging aspect of the algorithms is to identify the nodes in the eigenvalue and eigenvector graphs. Notice that with the exception of degenerate points, a node in an eigenvector graph or eigenvalue graph is a connected component of the domain that has certain characteristics, e.g., $W_{r,n}$, $W_{r,s}$, $W_{c,n}$, $W_{c,s}$ for the eigenvector graph, and D^+, D^-, R^+, R^-, and S for the eigenvalue graph. We refer to part of the domain corresponding to such a node as a *region*. Notice that a region can intersect multiple triangles or be contained completely inside one triangle. We refer to the intersection of a region with a triangle to be a *subregion*. A subregion is represented by the set of its boundaries. We represent a region as a list of all subregions.
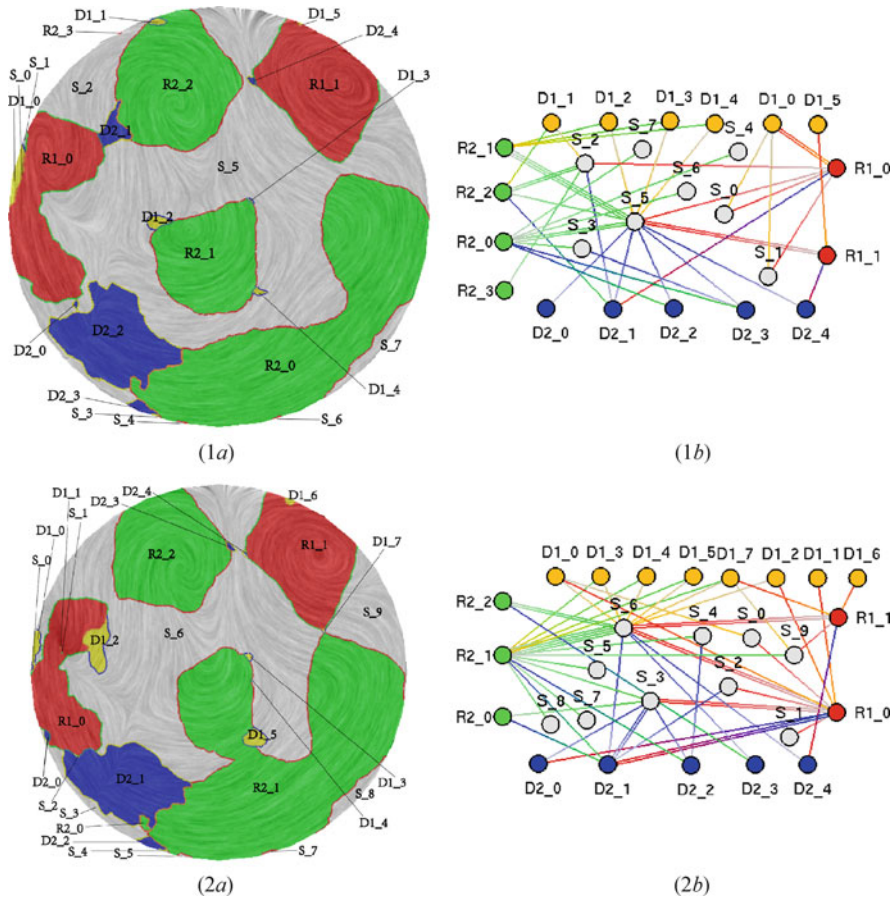
Fig. 4 This figure illustrates the eigenvalue graphs applied to slices of a 3D engine simulation data corresponding to those shown in Fig. 2. Each segment of the boundary between a pair of regions is depicted as an edge in the graph between the corresponding two nodes. For instance, there are two edges connecting node $R1\_1$ and S_5 in (1b), which indicate that the boundary between region $R1\_1$ and S_5 in (1a) is divided into two segments

To construct the eigenvector graph, we first compute the intersection of each triangle with a particular type of region, i.e., $W_{r,n}$, $W_{r,s}$, $W_{c,n}$, or $W_{c,s}$. This leads to a partition of each triangle into subregions of different types. Next, we iteratively merge adjacency subregions in adjacent triangles that belong to the same type. This leads to the regions, i.e., the nodes in the eigenvector graph. As part of the second step, we also establish adjacency relationship between regions of different types, thus constructing the edges in the graph. Finally, we extract the degenerate points which are also nodes in the graph. We then generate edges that connect these nodes with their container nodes, which must be a complex region.

The most complicated part of the computation is the first step, i.e., computing the intersection of a triangle with a particular type of region. This requires the extraction of the region boundaries, which satisfy the conditions $\gamma_r = 0$ (equator), $\gamma_s = \gamma_r$ (north 45° latitude), and $\gamma_s = -\gamma_r$ (south 45° latitude). Given the interpolation scheme, $\gamma_r = 0$ corresponds to linear segments while $\gamma_s = |\gamma_r|$ correspond(s) to quadratic curves. While all these types of boundaries can intersect the edges of a triangle, it is possible that an ellipse can be contained entirely inside a triangle. This last case corresponds to a $W_{c,n}$ or $W_{c,s}$ region strictly inside the triangle. Our algorithm is able to detect these cases as well. Figure 5 provides an illustration of this. Given a triangle (Fig. 5a), we perform the following computation:

1. For each edge in the triangle, decide whether and where it intersects the boundary of a particular type of a region (Fig. 5b).
2. Trace boundary curves within the triangle from the aforementioned intersection points on the edges of the triangle (Fig. 5c).
3. Determine if any internal elliptical region exists, and locate its position (Fig. 5d).
4. Create subregions and classify their types (Fig. 5e).

After all subregions have been computed, graph nodes are created for the subregions and a merging process is conducted to consolidate adjacent subregions with the same type into regions (Fig. 5f).

Constructing the eigenvalue graph is similar (Fig. 6). The only difference is that the boundaries of subregions can also intersect at junction points that are in the interior of a triangle. Consequently, we need to first locate all junction points (Fig. 6b) before tracing the boundaries (Fig. 6c).

Locate intersection points: An edge of a triangle can be parameterized by a parameter t where $0 \leq t \leq 1$. Denote the tensors at the two vertices of the edge as
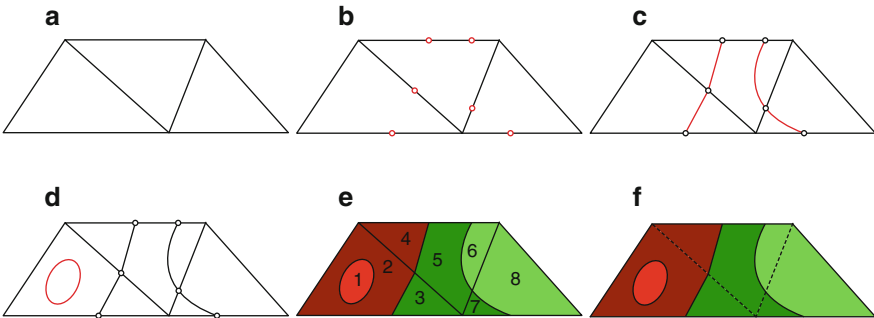
Fig. 5 Eigenvector graph construction: (**a**) given a tensor field defined on a triangular mesh, (**b**) each edge of the triangles is visited to locate possible intersections with region boundaries; (**c**) starting from the intersection points, the boundaries are traced within each triangle; (**d**) possible internal elliptical boundary is detected; (**e**) after subregions are created, their types are determined; (**f**) finally, a merging process is performed to consolidate adjacent subregions with same type into a single region
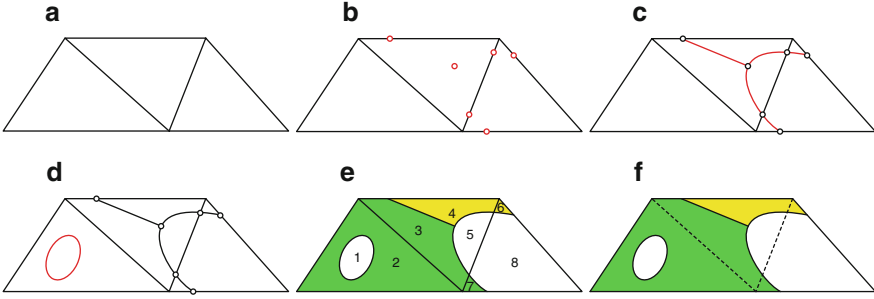
Fig. 6 Eigenvalue graph construction: (**a**) given a tensor field defined on a triangular mesh, (**b**) triangle edges are visited to locate possible intersections with region boundaries, possible junction points within the triangle are also detected; (**c**) the boundaries are then traced starting from the intersection points; (**d**) possible internal elliptical boundary is detected; (**e**) subregions are then created and their types are classified; (**f**) finally, adjacent subregions with same types are merged into a single region

$T_i = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}$ where $i = 1, 2$, then the tensor can be linearly interpolated for points on the edge as

$$T_t = \begin{pmatrix} a_t & b_t \\ c_t & d_t \end{pmatrix} = \begin{pmatrix} (1-t)a_1 + ta_2 & (1-t)b_1 + tb_2 \\ (1-t)c_1 + tc_2 & (1-t)d_1 + td_2 \end{pmatrix} \quad (2)$$

The points on the boundary between $W_{r,n}$ and $W_{r,s}$ regions satisfy $\gamma_r = 0$, i.e., $a_t + d_t = 0$, which is a linear equation. For points on the boundary between a real and complex domain pair satisfy $\gamma_s = |\gamma_r|$, which can be found by solving the following quadratic equation:

$$g(t) = \gamma_s|_t^2 - \gamma_r|_t^2 = \frac{(a_t - d_t)^2}{4} + \frac{(b_t + c_t)^2}{4} - \frac{(c_t - b_t)^2}{4} = 0 \quad (3)$$

Note that not all solutions are real intersection points. We will only accept those appearing on the intended edge segment.

Locate junction points: In the case of eigenvalue graph, there can be junction points located within the triangles, satisfying $|\gamma_d| = |\gamma_r| = \gamma_s$. We can find such points by identifying intersection points between two types of curves: the boundary between real and complex domains ($\gamma_s = |\gamma_r|$) and the boundary between scaling-dominant and rotation-dominant regions ($\gamma_d = \pm \gamma_r$). Recall that the former leads to a quadratic equation while the latter a linear one. There are a maximum of four solutions in every triangle, each of which corresponds to a junction point.

Trace boundary curves: Starting from the intersection points and the junction points, we trace the boundaries in each triangle. As mentioned above, the boundaries are either piecewise linear or piecewise quadratic.

The components of the tensor, i.e., a, b, c, and d, inside a triangle can be linearly interpolated using local coordinates (x, y). For each boundary, we first determine a scalar function such that the boundary is one of the function's iso-lines. For instance,

points on the boundary that separate real and complex regions satisfy $\gamma_s = |\gamma_r|$, i.e., $(a-d)^2 + (b+c)^2 - (c-b)^2 = 0$. We define the following scalar function:

$$g(x, y) = (a(x, y) - d(x, y))^2 + (b(x, y) + c(x, y))^2 - (c(x, y) - b(x, y))^2 \quad (4)$$

The tracing direction is given by the gradient of this function from the intersection or junction point.

To determine the direction at a given point during tracing, we rotate the gradient vector of $g(x, y)$ by $\pi/2$. Directions for other types of boundaries are obtained similarly. Each boundary is guaranteed to end at an intersection point or a junction point (Fig. 6c).

Internal elliptical boundaries: As previously mentioned, a region may be completely contained in a triangle. When this happens, the region boundary must be elliptical. Such an ellipse has no intersections with triangle edges or another region boundary except in degenerate cases, and cannot be identified by tracing from intersection points or junction points.

To detect such an ellipse, we first compute the coefficients in (4) for the quadratic boundary and determine whether it is an ellipse or a hyperbola. If it is an ellipse, there can be four scenarios:

1. The ellipse intersects with the triangle.
2. The ellipse is entirely inside the triangle.
3. The ellipse is entirely outside of the triangle and has no intersections with the triangle.
4. The ellipse encloses the triangle.

If the ellipse does not intersect the triangle, we evaluate the quadratic function from (4) at the center of the ellipse and the vertices of the triangle. Based on the sign of these values as well as whether the center of the ellipse is inside the triangle, we can determine whether an internal ellipse exists, and if so create a subregion (Fig. 5d).

4   Results

We apply our graphs to a diesel-engine simulation data set. The asymmetric tensor field we analyze is the velocity gradient tensor field. Figures 2 and 4 show the eigenvector and eigenvalue graphs, respectively, for two slices (10 mm apart). In both figures the textures in the background illustrate the input vector field.

Figure 7 uses another slice (83 mm from the top of the engine cylinder) to compare the eigenvector graph and the eigenvalue graph with the entity connection graph (ECG) [2], a more general form of vector field topology than vector field skeleton. ECG highlights the connectivity among the fixed points via separatrices, which represents quite different topology from those of the tensor topology. We believe

(1a)

(1b)

(2a)

(2b)

(3a)

(3b)
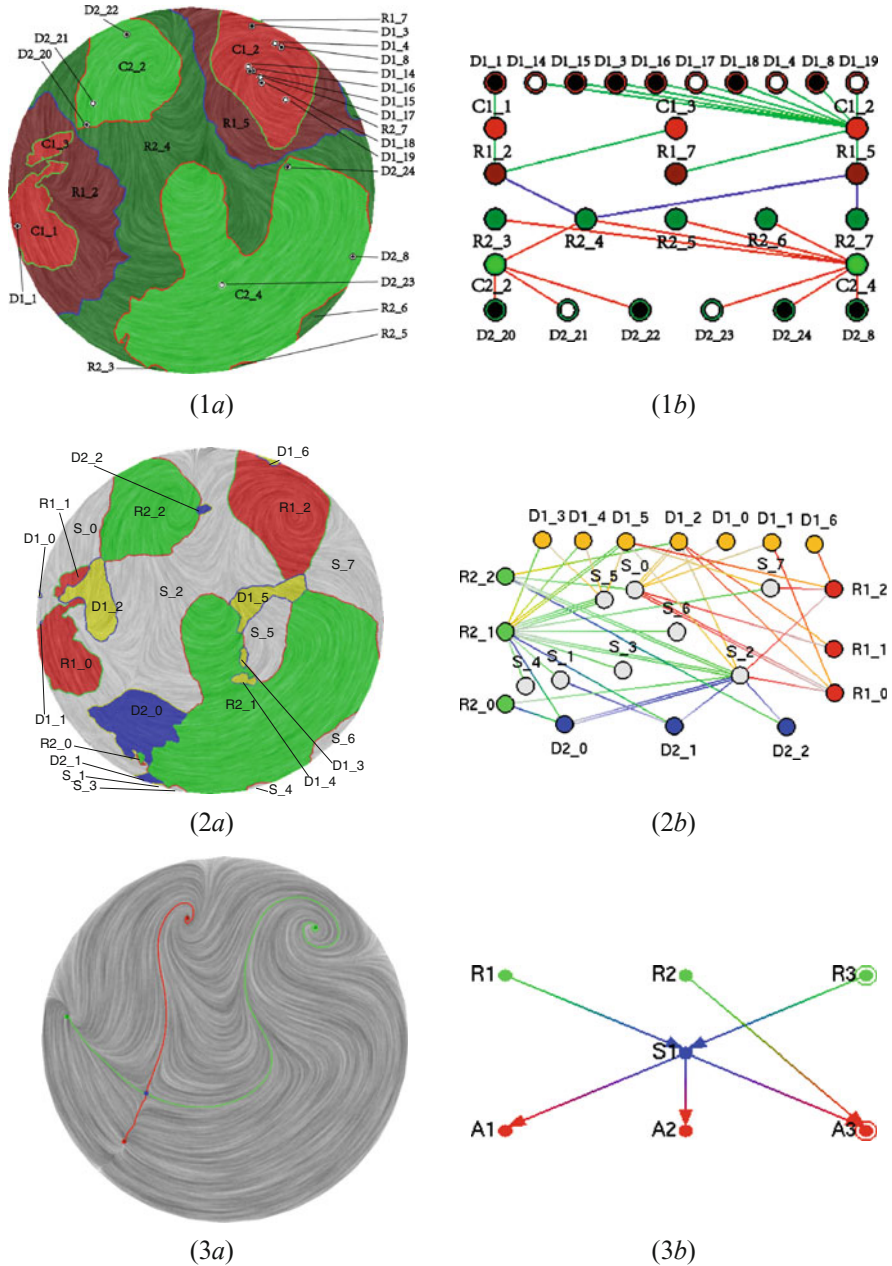
Fig. 7 This figure compares the eigenvector graph (1), eigenvalue graph (2), and vector field topology (3) of the slice at 83 mm from the top of the cylinder in the diesel engine simulation. In (3b), *R* and *A* represent the repellers and attractors, and *S* indicates the saddles. Vector field topology and tensor field topology can provide mutually complemental information of the data

that vector field topology and tensor field topology can provide complemental information about the underlying flow.

The eigenvector graphs in Fig. 2 demonstrate how and where a pair of co-rotating vortices coalesce into a single vortex of the same rotation. This can be seen by examining multiple nodes in the complex domain (light green or light red) of the same hemisphere that are connected to a common node in the real domain (dark green or dark red) of the same hemisphere. A couple of vortex coalescence processes can be detected in the transition from (1) to (2): see $C2\_1$ and $C2\_2$, and $C2\_3$ and $C2\_4$ in Fig. 2. On the other hand, a pair of counter-rotating vortices tends to maintain its form. A counter-rotating vortex pair can be identified by the edge connection from a node in the complex domain (light green or light red) to the corresponding node in the complex domain in the opposite side (light red or light green). The connection needs to go through two nodes of the different rotations in the real domain. It must be emphasized that the foregoing vortex behaviors are often discussed for line vortices in fluid mechanics. Our tensor field topology enables the users to analyze complex flows with basic understanding in fluid mechanics. The eigenvalue graphs shown in Fig. 4 can provide systematic criteria for flow feature reduction, i.e., part of the multi-scale analysis. For example, an isolated node in the graph that is connected with only one edge may merge into the surrounding flow characteristic, such as S_1 and $D1\_1$ in Fig. 4(2b).

5   Conclusion and Future Work

We have defined the topology of asymmetric tensor fields in terms of *eigenvector graphs* and *eigenvalue graphs*, and introduced algorithms to construct the graphs given a tensor field. We also point out several topological properties for asymmetric tensor field topology and their physical meanings in terms of fluid mechanics.

In future research, we plan to study how to combine the three types of graphs, i.e., eigenvalue graph, eigenvector graph, and entity connection graph (vector field topology), in order to provide stronger analysis of the vector fields. The quality of a graph is greatly effected by its layout. We plan to explore optimal graph layout algorithms for eigenvalue and eigenvector graphs. Compared to vector fields and symmetric tensor fields, the topological changes for time-dependent asymmetric tensor fields appear to be more complex and require more sophisticated techniques for feature tracking. In addition, it is less intuitive how to perform asymmetric tensor field simplification which is important to multi-scale analysis. We plan to address these challenges in our future research. Finally, it is our goal to extend this work to the topological analysis of 3D asymmetric tensor fields, a largely untouched topic in tensor field visualization.

References

1. Bremer, P.T., Edelsbrunner, H., Hamann, B., Pascucci, V.: A topological hierarchy for functions on triangulated surfaces. IEEE Trans. Visual. Comput. Graph. **10**(4), 385–396 (2004)
2. Chen, G., Mischaikow, K., Laramee, R.S., Pilarczyk, P., Zhang, E.: Vector field editing and periodic orbit extraction using morse decomposition. IEEE Trans. Visual. Comput. Graph. **13**(4), 769–785 (2007)
3. Delmarcelle, T., Hesselink, L.: Visualizing second-order tensor fields with hyperstream lines. IEEE Comput. Graph. Appl. **13**(4), 25–33 (1993)
4. Delmarcelle, T., Hesselink, L.: The topology of symmetric, second-order tensor fields. In: Proceedings IEEE Visualization '94, Washington, DC, USA (1994)
5. Dodd, R.K.: A new approach to the visualization of tensor fields. Graph. Model. Image Process. **60**(4), 286–303 (1998)
6. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical morse complexes for piecewise linear 2-manifolds. In: SCG '01: Proceedings of the 17th Annual Symposium on Computational Geometry, pp. 70–79. ACM, New York, NY, USA (2001). doi: http://doi.acm.org/10.1145/378583.378626
7. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
8. Helman, J.L., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. IEEE Computer **22**(8), 27–36 (1989)
9. Hesselink, L., Levy, Y., Lavin, Y.: The topology of symmetric, second-order 3d tensor fields. IEEE Trans. Visual. Comput. Graph. **3**(1), 1–11 (1997)
10. Laramee, R.S., Garth, C., Schneider, J., Hauser, H.: Texture-advection on stream surfaces: a novel hybrid visualization applied to cfd results. In: Data Visualization, The Joint Eurographics—IEEE VGTC Symposium on Visualization (EuroVis 2006), Lisbon, Portugal, pp. 155–162 (2006)
11. Laramee, R.S., Hauser, H., Zhao, L., Post, F.H.: Topology-based flow visualization: the state of the art. In: The Topology-Based Methods in Visualization Workshop (TopoInVis 2005), Visualization and Mathematics, Budmerice, Slovakia, pp. 1–19 (2007)
12. Laramee, R.S., Weiskopf, D., Schneider, J., Hauser, H.: Investigating swirl and tumble flow with a comparison of visualization techniques. In: Proceedings IEEE Visualization 2004, Austin, TX, USA, pp. 51–58 (2004)
13. Scheuermann, G., Krüger, H., Menzel, M., Rockwood, A.P.: Visualizing nonlinear vector field topology. IEEE Trans. Visual. Comput. Graph. **4**(2), 109–116 (1998)
14. Tricoche, X., Scheuermann, G.: Continuous topology simplification of planar vector fields. In: Proceedings IEEE Visualization 2001, San Diego, CA, USA, pp. 159–166 (2001)
15. Tricoche, X., Scheuermann, G., Hagen, H.: A topology simplification method for 2d vector fields. In: Proceedings IEEE Visualization 2000, Salt Lake City, UT, USA (2000)
16. Tricoche, X., Scheuermann, G., Hagen, H.: Scaling the topology of symmetric, second-order planar tensor fields. In: Proceedings of NSF/DOE Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization, Lake Tahoe, CA, USA (2001)
17. Tricoche, X., Scheuermann, G., Hagen, H.: Topology Simplification of Symmetric, Second-Order 2D Tensor Fields, Hierarchical and Geometrical Methods in Scientific Visualization. Springer, Heidelberg (2003)
18. Tricoche, X., Wischgoll, T., Scheuermann, G., Hagen, H.: Topology tracking for the visualization of time-dependent two-dimensional flows. Comput. Graph. **26**(2), 249–257 (2002)

19. Wischgoll, T., Scheuermann, G.: Detection and visualization of closed streamlines in planar fields. IEEE Trans. Visual. Comput. Graph. **7**(2), 165–172 (2001)
20. Wischgoll, T., Scheuermann, G.: Locating Closed streamlines in 3D vector fields. In: Proceedings of the Joint Eurographics—IEEE TCVG Symposium on Visualization (VisSym 02), saarbrücken, Germany, pp. 227–280 (2002)
21. Zhang, E., Hays, J., Turk, G.: Interactive tensor field design and visualization on surfaces. IEEE Trans. Visual. Comput. Graph. **13**(1), 94–107 (2007)
22. Zhang, E., Mischaikow, K., Turk, G.: Vector field design on surfaces. ACM Trans. Graph. **25**(4), 1294–1326 (2006)
23. Zhang, E., Yeh, H., Lin, Z., Laramee, R.S.: Asymmetric tensor analysis for flow visualization. IEEE Trans. Visual. Comput. Graph. **15**(1), 106–122 (2009)
24. Zheng, X., Pang, A.: 2D asymmetric tensor analysis. In: IEEE Proceedings on Visualization, pp. 3–10 (2005)
25. Zheng, X., Parlett, B., Pang, A.: Topological lines in 3D tensor fields and discriminant Hessian factorization. IEEE Trans. Visual. Comput. Graph. **11**(4), 395–407 (2005)

Part IV
Topological Visualization of unsteady flow

On the Elusive Concept of Lagrangian Coherent Structures

Jens Kasten, Ingrid Hotz, and Hans-Christian Hege

1 Introduction

The concept of *Lagrangian coherent structures* (LCS) plays a fundamental role in the analysis of time-dependent flow fields. The idea of *coherent structures* (CS) developed about fifty years ago in the context of semi- or fully turbulent flows [11]. One of the first observations of such large-scale structures was made by Roshko and Brown [1] at a turbulent plane mixing layer using shadowgraphs. Before CS were found, it was assumed that turbulent flows are only determined by chaotic particle motion where structures of different scales arise, dissolve and affect each other. The analysis of the vast amount of structures was restricted to statistical methods. Today, there is a general consensus that besides the chaotic motion of particles, there are CS exhibiting a more coarse and orderly flow behavior. The finding that turbulent flow is not purely chaotic but embodies orderly structures had a deep impact on fluid mechanics.

However, despite their importance, there is no commonly accepted precise mathematical definition of CS. Moreover, Hussain [12] states that "in principle, concepts like CS are best left implicit". On the other side, the extraction of CS is a fundamental goal in the analysis of complex flow fields. As a result, a variety of individual interpretations and more or less formal definitions of CS have been proposed, from an Eulerian as well as Lagrangian point of view. Examples are the definitions by Michalke [12] or Farge [3], both proposing a "coherence function". Others give more conceptual definitions. Yule suggests that CS should follow three requirements: (1) being repetitive, (2) survive distances larger than structure size, (3) significantly contribute to the kinetic energy [30]. While Hussain considers these conditions, at least partly, as "unnecessary and unrealistic", he defines CS as

J. Kasten (✉) · I.Hotz · H.-C. Hege
Zuse Institute Berlin (ZIB), Takustraße 7, 14195 Berlin, Germany
e-mail: kasten@zib.de; hotz@zib.de; hege@zib.de

"connected turbulent fluid mass with instantaneously phase-correlated vorticity over its spatial extent" [11]. Farge states, that "the only definition of a coherent structure that seems objective is a locally meta-stable state, such that, in the reference frame associated with the coherent structure, the nonlinearity of Navier Stokes equations becomes negligible" [3].

In the past ten years, an approach to LCS by Haller became quite popular. He proposes a concept of distinguished material lines or surfaces in the flow field. To extract these structures, he refers to the finite-time Lyapunov exponent (FTLE) [9]. FTLE measures the separation of infinitesimally close particles over a finite-time interval T. Typically, FTLE is computed using the gradient of the flow map. Haller states that the extremal structures of the FTLE field are possible boundaries of LCS [10]. Shadden and Marsden [23, 24] even define ridges of the resulting scalar field as LCS.

More recently, other Lagrangian feature extraction methods dealing with the eduction of LCS have been investigated in the visualization community. Typically, a local feature identifier is integrated along a pathline resulting in an averaged scalar value. The finite time span T used for the averaging is a crucial parameter of all these methods. It defines the temporal scale of the feature. Examples for local feature identifiers are acceleration, separation or unsteadiness [4, 13, 14].

The purpose of this paper is to critically review selected results of popular realizations of LCS related to FTLE. This comprises raising questions as whether Lagrangian features and their development should always be associated with path-lines. It is not intended to give final answers but to illustrate various aspects of LCS. As an addition to existing feature concepts, we propose a formal complementary Lagrangian feature concept that possibly fills some gaps. It includes the concept of a feature identifier equipped with two further components: a spatial and a temporal measure for the importance of the local structure, *feature strength* and *feature lifetime*.

For the analysis, we will use some two-dimensional well-known datasets. The Stuart vortex model [25] illustrates the flow of two layers of fluid with different velocities. The Oseen vortex model [18] is a simple analytical model that is complex enough to generate different time-dependent features. Furthermore, the flow behind a circular cylinder is used to discuss known FTLE structures. As an outlook, we use the dataset of a jet that is a complex turbulent dataset.

2 Concepts of Lagrangian Coherent Structures

Currently, many efforts in visualization and analysis of unsteady flows take up the idea of LCS. The goal of this section is to recall some of the reoccurring ideas and notions and to try to distill an underlying concept.

2.1 Material Lines

Many discussions pursue the idea that temporally developing flow features should be close to material lines. Such features result in structures building a kind of material barrier, minimizing crossflow.

Haller defines a material line as a "smooth curve of fluid particles advected by the velocity field" and describes them as finite-time invariant manifolds [9]. In terms of integral surfaces, material surfaces are path surfaces and material lines correspond to time lines in the surface. Material lines can be interpreted as transport barrier. In this context, Haller proposes to define LCS boundaries as distinguished material lines advected with the flow [8]. As an example, he refers to hyperbolic repelling material lines.

2.2 Pathline Predicates

Independently, Salzbrunn et al. [22] proposed a framework based on pathline predicates. A Boolean function decides if a pathline, constricted to the data domain, exhibits a certain property of interest. Thereby, two groups of predicates can be distinguished. The first group considers the pathline as geometric entity. The second group relies on properties derived from the flow like vorticity. This framework is very general. Many pathline-based approaches, including material surfaces, can be subsumed under this concept. Methods that do not investigate the entire available pathline, but only a finite-time interval T, do not strictly follow this idea of pathline predicates. This comprises approaches that average local feature identifiers along pathlines, like unsteadiness and acceleration [4, 13].

2.3 Finite-time Lyapunov Exponent (FTLE)

Currently, the most common realizations of LCS in flow visualization are based on FTLE. It measures the separation of infinitesimally close particles over a finite-time interval T and became popular through Haller's work. He has suggested characterizing LCS as extremal structures of both the forward and backward FTLE field. Later on, he has relativized this view presenting an example where FTLE ridges do not necessary mark LCS [10]. But still, FTLE ridges depict LCS in a wide range of configurations and are used in many applications [19]. Shadden even defines LCS as ridges in FTLE. He shows that the resulting structures are approximately advected by the flow with negligible fluxes across the structures [23]. This puts FTLE into context with material lines and surfaces.

There are multiple possibilities to implement the concept of FTLE. Typically, the computation of FTLE utilizes the flowmap (F-FTLE). Since this involves the integration of multiple pathlines, it does not strictly fit into the framework of pathline predicates. Another variant, called *localized FTLE* (L-FTLE), bases the computation on the local separation generated by the Jacobian along a single pathline. This approach comes closer to the concept of pathline predicates.

2.4 In Summary

Most methods are characterized by a local feature identifier (separation, acceleration, pathline curvature, unsteadiness) that is accumulated or averaged for a given finite time T along a pathline. Differences origin from different time parameters T and different local measures. The proper choice of the parameter T is subject of many discussions. In general, an increasing time parameter T is considered to lead to more accurate results.

In the following, we examine the two major underlying assumptions in more detail: 1. Longer integration times lead to more accurate results. 2. Features are attached to pathlines. Thereby we focus on local identifiers related to stretching, but many of the arguments carry over to the more general case.

3   Separation Related Feature Identifiers

We base our investigations on two feature identifiers related to flow separation and convergence, which will be introduced in this section. The first one is the FTLE. In this paper, we have chosen the L-FTLE variant. For the examples considered, the flow map approach yields very similar results and, thus, this choice is not essential for the following discussion. A detailed comparison of both methods can be found in [14]. As second measure, we introduce *averaged stretching*, which only considers the local norm of the separation without taking its directions into account. There are other interesting FTLE-related measures as proposed by Haller [9], which are not considered here.

3.1 L-FTLE

While the flowmap FTLE (F-FTLE) measures the separation of close-by particles over a finite period of time, L-FTLE reflects the behavior of infinitesimally close particles along a single pathline. It is based on the integration of the Jacobian matrix J along the pathline. Given a flow field v, the Jacobian of v is a generator of separation. Its symmetrical part quantifies the separation along the pathline.

Consider a pathline $p(t) = p(x_0, t_0, t)$ for a particle started at space-time location (x_0, t_0). The deviation of trajectories of infinitesimally close particles started at $(x_0 + \delta_0, t_0)$, with $\delta_0 \to 0$, is given by the differential equation

$$\dot{\delta}(t) = J(p(t), t_0 + t)\delta(t). \tag{1}$$

Solving the differential equation yields

$$\delta(t) = exp\left(\int_0^t J(p(\tau), t_0 + \tau)d\tau \right)\delta_0. \tag{2}$$

Given a finite time span T, the matrix

$$\Psi_T(p) = exp\left(\int_0^T J(p(t), t_0 + t)dt \right) \tag{3}$$

expresses the mapping of a neighborhood at the starting point $p(0)$ onto its deviations at the end point $p(T)$. Compared to the flow map approach, this matrix corresponds to the gradient of the flow map. Defining a temporal discretization of the pathline $T = \Delta t \cdot N$, where Δt is the length of one time step and N the number of steps, Equation 3 can be approximated as

$$\Psi_T(p) \simeq \prod_{i=0}^{N-1} exp(J(p(i\,\Delta t), t_0 + i\,\Delta t) \cdot \Delta t). \tag{4}$$

L-FTLE is now defined as the largest separation of this mapping. It is computed as

$$\text{L-FTLE}^+(\mathbf{x}_0, t_0, T) = \frac{1}{T} \ln(||\Psi_T(p)||_\lambda), \tag{5}$$

where $||.||_\lambda$ depicts the spectral norm of the resulting matrix.

3.2 Averaged Stretching

While L-FTLE considers the deformation of the neighborhood of a particle over a finite time T, we are now interested in the averaged separation a particle experiences, independent from its direction. This corresponds to the average of the instantaneous maximum separation. In the discretized version, we therefore use the L-FTLE computed only for one time step Δt and define

$$\mathscr{S}_{\Delta t}(\mathbf{x}, t) = \frac{1}{\Delta t} \ln\left(|| \exp(J(\mathbf{x}, t)\Delta t)) ||_\lambda \right). \tag{6}$$

This measure is averaged along the pathline for a finite time interval $T = \Delta t \cdot N$

$$\mathscr{I}_T(\mathbf{x}_0, t_0) = \frac{1}{T} \sum_{k=0}^{N} \mathscr{S}_{\Delta t}(p(\mathbf{x}_0, t_0, k\Delta t), t_0 + k \cdot \Delta t) \cdot \Delta t. \qquad (7)$$

4   Critical Analysis of Selected Examples

In this section, the measures introduced in Sect. 3 are applied to selected examples. Thereby, we will analyze the resulting structures and put it into context with the notion of LCS and pathline related features.

4.1   Stuart Vortices

The analytic Stuart vortex model represents a two-dimensional time-dependent flow with elliptical convecting vortices moving from left to right with the medium velocity of the flow field. The model can be interpreted as a simple version of a shear layer, i.e., the velocity is lower in the upper half than in the lower half. The flow is analytically defined as

$$u(\mathbf{x}, t) = \frac{\sinh(2\,x_2)}{\cosh(2\,x_2) - 0.25 \cdot \cos(2(x_1 - t))} + 1$$

$$v(\mathbf{x}, t) = -\frac{0.25 \cdot \sin(2(x_1 - t))}{\cosh(2\,x_2) - 0.25 \cdot \cos(2(x_1 - t))}$$

(8)

It is a periodic flow with a temporal period of $T = \pi$. We use a dataset resulting from a sampling to a regular grid.

To analyze the dataset, we computed the forward L-FTLE for different integration times T, see Fig. 1. Red indicates high FTLE values and white low values. The blue regions depict areas, where the FTLE cannot be computed due to the dataset boundary. As expected, the increasing integration time leads to more and crisper details in the visualization. The elliptical Stuart vortices are clearly highlighted.

Besides, one can make following observations: After a few periods a stable ridge bounding the vortices is emerging. This matches the statement by Green et al. [7] that the location of the ridge indicating the boundary of the vortex does not change with increasing time. In contrast, the inner regions become clearer and sharper, but also change in position and frequency. The location of these FTLE-ridges depends on the integration time and does not converge for increasing integration time T. The high FTLE values result from "separation events" far away in spatio-temporal

domain, which are advected with the flow. The significance of such features for flow analysis is not clear and should not be interpreted as LCS.

In order to filter out the stable ridges, representing LCS, we propose to average the FTLE fields belonging to different integration times. The result for an average of integration times $T = 10, 11$ and 12 is shown at the bottom right of Fig. 1. The high FTLE values within the vortex vanish and the region boundaries stay visible.

Next, we focus on the white regions in the center of the vortices indicating no separation. Here, the FTLE value changes from red (high separation) for small T to white (low separation) after a few integration periods. In the center, the locally strong separation vanishes. To understand this behavior, we have seeded three pathlines, one in the vortex center and two on the vortex boundary on the FTLE ridge. Along the pathline, the instantaneous deformation induced by the Jacobian $\exp(J(\mathbf{x}, t)\Delta t)$ is visualized using glyphs, see Fig. 2. In the left figure, the seeding points are displayed including the first glyph. The right image shows the pathlines and the glyphs in a three-dimensional visualization, where time is used as third dimension. The integration time for the FTLE and the pathlines is $T = 10$, which corresponds to approximately three periods of the dataset. The color of the glyphs is determined by the maximum local separation as defined in Equation 6. Blue depicts high and white low values. It can be seen that the pathline in the center exhibits a
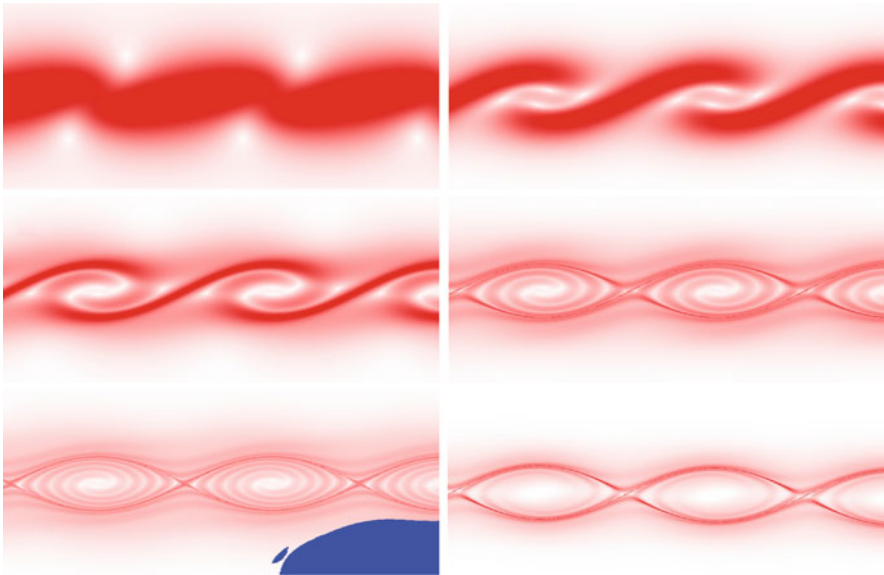
Fig. 1 Forward L-FTLE of the Stuart vortex dataset for $T = 1, 2, 5, 10$ and 15, where one period corresponds to $T = \pi$ (from left top to bottom right). The image bottom right is the result of averaging the images for $T = 10, 11$ and 12. The colormap is the same for all images – white color corresponds to low and red to high FTLE values. With the increasing integration time the features become more crisp and detailed, partially resulting from an structure advection along trajectories
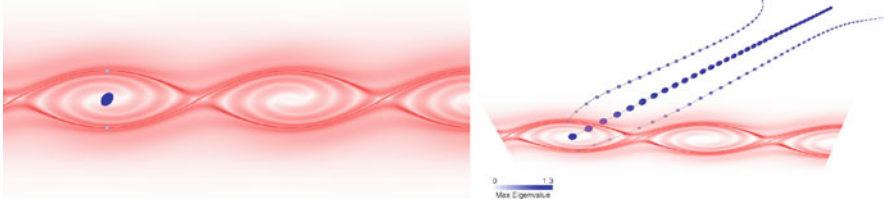
Fig. 2 Analysis of the local separation along three pathlines. The colormap is the same as in Fig. 1. The local separation is depicted by glyphs deformed by the Jacobian matrix. While the FTLE value is high in the outer regions and low in the center, the local separation shows strong values along the center pathline and lower values along the outer pathlines
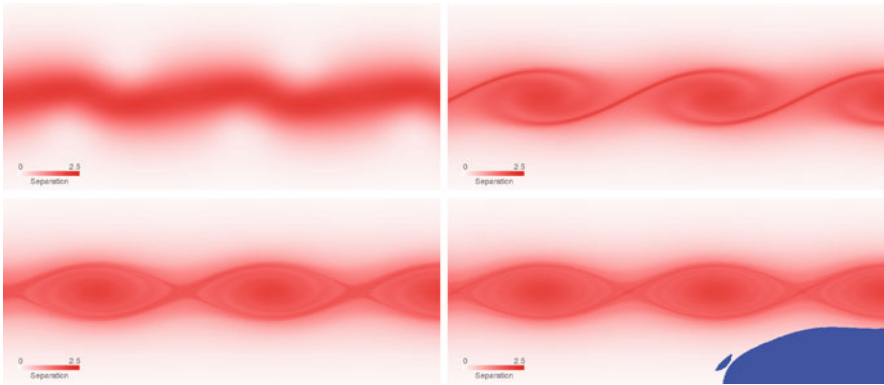
Fig. 3 Forward averaged stretching values of the Stuart vortex dataset for $T = 1, 5, 10$ and 15 (from left top to bottom right). With increasing integration times, the separation in the vortex centers does not vanish in contrast to the values of the L-FTLE measure

constantly high local separation, which is always higher than the local separation along the other two pathlines. Moreover, the separation points always in the same direction on the center pathline. Thus, it might be surprising at first sight that the FTLE value vanishes. Closer inspection shows that this results from the rotational part of the Jacobian, which is not visualized in this representation. Keeping this in mind, it cannot be concluded that low FTLE values mean that the particle does not experience high separation. FTLE shows a combination of both flow components – rotation and stretch.

A feature identifier that measures the local separation a particle experiences has been introduced in Equation 6. Figure 3 displays the result for integration times of $T = 1, 5, 10$ and 15. Red marks high and white low average separation. Again blue regions depict seedings where the pathlines left the domain. These visualizations illustrate that the average separation is high in the entire vortex region with maximal values in the center and at the boundary of the vortices. The averaged separation value converges toward its final value after a few time periods.

To put it in a nutshell, we can make the following observations when analyzing this dataset:

1. The locations of FTLE ridges do not necessarily indicate boundaries of LCS. One has to distinguish two kinds of structures. First, there are structures that stabilize after some integration periods. The corresponding pathlines experience periodic "separation events" and represent the actual boundaries of LCS. Second, there are structures with no definite position, where one "separation event" is transported along pathlines. It is to expect that for an integration time approaching infinity, it will become arbitrary dense and will fade out.
2. Low FTLE values do not necessarily mean that there is no separation along the pathline. Therefore, it is meaningful to analyze the local separation in addition to FTLE.

4.2   Cylinder Dataset

This dataset, referred to as *cylinder* dataset, resulted from a time-dependent 2D CFD simulation of the von Kármán vortex street [17, 29], the flow behind a cylinder with $Re = 100$, see Fig. 4. It consists of 32 time steps. The flow is periodic, allowing a temporally unbounded evaluation of the field.

To analyze the contribution of the advection of separation to the final FTLE image, we compare the FTLE results, cf. Fig. 5, with a simple visualization using dye-advection, cf. Fig. 4. The dye is injected into the time-dependent flow in front of the cylinder. Advection results in a streak line visualization with very similar patterns to backward FTLE. Thus, if local separation takes place constantly at one position – as it is the case for this dataset at the cylinder –, FTLE mainly transports this high separation values just as streaklines transport the stationary dye. The apparent feature originates mainly from the local separation at the cylinder.
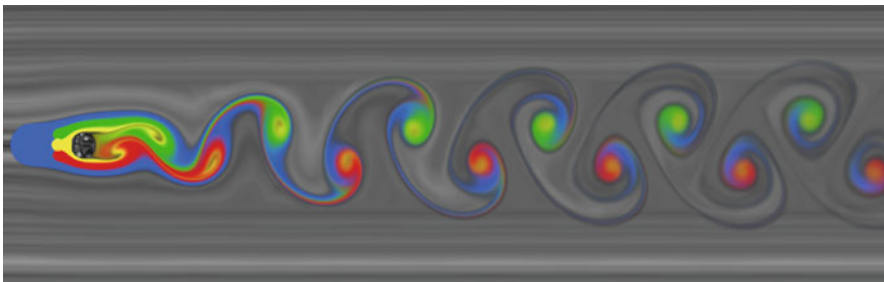
Fig. 4 Visualization of time-dependent dye-advection combined with image-based flow visualization. The dye is advected along streaklines. The typical pattern of backward FTLE can be recognized in the advected dye

Fig. 5 L-FTLE computed for the cylinder dataset computed for two shedding periods. White *color* corresponds to low and *blue* to high FTLE values

4.3 Mixing of Oseen Vortices

The Oseen vortex models a line vortex that decays due to viscosity. The velocity V_Θ in the circumferential direction θ is given by

$$V_\Theta(r) = \frac{\Gamma}{2\pi} \frac{1 - e^{-(\frac{r}{r_c})^2}}{r}, \tag{9}$$

where r is the spatial coordinate with origin in the center of the vortex, r_c is a parameter determining the core radius and the parameter Γ the circulation contained in the vortex. For further information we refer to Rom-Kedar et al. [21] or Noack et al. [16].

We use this example to discuss the second assumption, that features ought to be attached to pathlines. For this purpose, we focus on saddle like features, since vortices have the property to trap particles inside. Figure 6 shows a time-dependent saddle-line, which is extracted as tracked minima of the acceleration. In addition, we seeded pathlines in the vicinity of this feature. As discussed in [13], the time-dependent saddle-line is not associated to any pathline. In general, particles do not stay in the vicinity of such features for a long time.

4.4 Turbulent Jet

This dataset represents a turbulent, non-periodic flow [2, 15]. It is generated from a three-dimensional time dependent large eddy simulation of a jet. It consists of about 6,000 timesteps – we used steps 5,000–5,500 for our computation, which corresponds to a temporal interval of length 50. From this dataset, the center slices are extracted resulting in a two-dimensional unsteady dataset. In Fig. 7, the forward and backward FTLE computed in the unsteady two-dimensional dataset for a length of $T = 5$ are displayed.
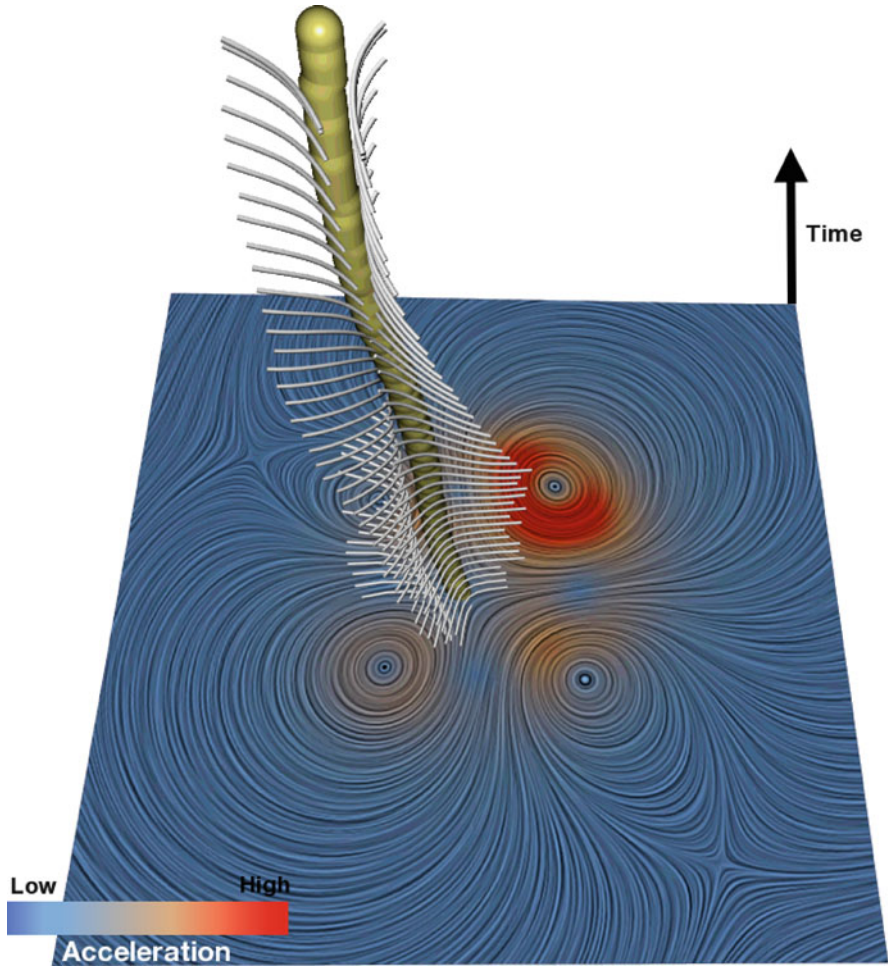
Fig. 6 2D time-dependent dataset of the mixing of six Oseen vortices with time as the third dimension. The LIC at the bottom is colored according to acceleration. The thick yellow line depicts tracked minima of the acceleration showing saddle behavior. In the vicinity, pathlines are seeded

The discovery of the existence of large-scale structures, so-called CS, superimposing a chaotic background-flow was a big breakthrough in turbulent flow research. Under this perspective, it is interesting to have a look on how well FTLE copes with turbulent data. Figure 7 shows the forward and backward FTLE structures extracted from the jet dataset. There is a vast amount of structures visible, in particular multiple crossings of forward and backward FTLE. Even though the explicit physical interpretation of this slicing is not clear it is sufficient to demonstrate the complexity of the emerging structures, which we leave without interpretation at this point.
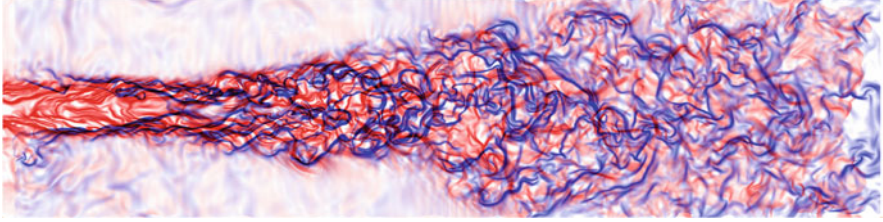
Fig. 7 Forward (*red*) and backward (*blue*) FTLE for the two-dimensional time-dependent dataset of a jet for the integration time $T = 5$. The colormap was introduced by Garth et al. [5]

5 An Alternative Concept of Features

The third example in our discussions suggests that there are also relevant features that do not fall into the framework of pathline predicates or can be interpreted as distinguished pathline. In the following, we propose an alternative point of view that considers features as spatio-temporal entities detached from single particles.

The idea is also based on a *local feature identifier* $\mathscr{F}(\mathbf{x}, t)$ similar to those accumulated or averaged to define pathline predicates. Instead of following these scalar values along the pathline, the resulting scalar field is examined for a fixed time step, e.g., by extracting its extremal structures. The Lagrangian perspective comes into play by choosing a Lagrangian identifier as, e.g., particle acceleration. Minima of the acceleration or maxima of the local separation correspond well to certain flow features as, e.g., vortices.

The temporal development of these features is no longer related to a particle trajectory and has to be tracked by other means. There are several concepts and algorithms available to solve similar tasks, e.g., Feature Flow Fields [26, 28] or the tracking of vector field singularities [6, 27]. Another tracking algorithm proposed by Reininghaus et al. [20] also addresses the problem of high feature density. The paper proposes an importance filter, which combines persistence as a *spatial feature strength* with the idea of *feature lifetime* [13]. This results in a spatio-temporal importance measure by integrating the spatial measure over the feature line. Similar filters could be used to distinguish important structures from non-relevant features.

6 Conclusions

This paper gives a short overview on the development and characteristics of LCSs in the area of flow visualization and analysis. In the history of the analysis of turbulent flows, LCS were a breakthrough, but, still, there is no common understanding nor formal definition, covering all aspects of LCS. The FTLE measure and its extremal structures can be considered as one successful realization of LCS. It highlights interesting flow structures in many applications. But one should also be aware that,

even in non-turbulent flow fields, there are structures that cannot be depicted by FTLE. It is important to interpret the results carefully to avoid misconceptions. There are also examples where similar features can be obtained with much simpler, less computationally expensive methods. As Example 4.2 shows, there can be a close connection to streaklines.

From the analysis of path-line related methods applied to a few well-known examples we summarize our observations as follows: Increasing integration time does not always improve the results. There are Lagrangian features that cannot be formulated in terms of pathline features. There is not yet a satisfying interpretation of FTLE ridges for complex and especially for non-periodic data sets. Finally, we conclude that FTLE is *one* realization of LCS but does not cover all aspects of LCS. There is also a need for features not related to pathlines, which could build on the concept of local identifiers, feature strength and lifetime.

References

1. Brown, G.L., Roshko, A.: On density effects and large structures in turbulent mixing layers. J. Fluid Mech., **64**(4), 775–816 (1974)
2. Comte, P., Dubief, Y., Bruni, C., Meinke, M., Schultz, C., Rister, T.: Simulation of spatially developing plane and round jets. Notes Numer. Fluid Mech., **66**, 301–319 (1998)
3. Farge, M., Schneider, K., Kevlahan, N.K. R.: Coherent structure eduction in wavelet-forced two-dimensional turbulent flows. In: Krause, E., (ed.) Proceedings of IUTAM Symposium on Dynamics of Slender Vortices (Aachen) (1998)
4. Fuchs, R., Kemmler, J., Schindler, B., Waser, J., Sadlo, F., Hauser, H., Peikert, R.: Toward a Lagrangian vector field topology. Comput. Graph. Forum, **29**(3), pp. 1163–1172 (2010)
5. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Vis. Comput. Graph., **13**, 1464–1471 (2007)
6. Garth, C., Tricoche, X., Scheuermann, G.: Tracking of vector field singularities in unstructured 3d time-dependent datasets. In: Proceedings of IEEE Visualization 2004, pp. 329–336 (2004)
7. Green, M.A., Rowley, C.W., Haller, G.: Detection of Lagrangian coherent structures in 3d turbulence. J. Fluid Mech., **572**, 111–112 (2007)
8. Haller, G.: Finding finite-time invariant manifolds in two-dimensional velocity fields. Chaos, **10**, 99–108 (2000)
9. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Physica D, **149**(4), 248–277 (2001)
10. Haller, G.: Lagrangian coherent structures from approximate velocity data. Phys. Fluids, **14**, 1851–1861 (2002)
11. Hussain, A.K.M.F.: Coherent structures – reality and myth. Phys. Fluids, **26**, 2816–2850 (1983)
12. Hussain, A.K.M.F.: Coherent structures and turbulence. J. Fluid Mech., **173**, 303–356 (1986)
13. Kasten, J., Hotz, I., Noack, B.R., Hege, H.-C.: On the extraction of long-living features in unsteady fluid flows. In: Pascucci et al., V. (ed.) Topological Methods in Data Analysis and

Visualization. Theory, Algorithms, and Applications, Mathematics and Visualization, Berlin, pp. 115–126. Springer (2010)

14. Kasten, J., Petz, C., Hotz, I., Noack, B.R., Hege, H.-C.: Localized finite-time Lyapunov exponent for unsteady flow analysis. In: Magnor et al, M. (ed.) Proceedings of Vision, Modeling, and Visualization (VMV'09), vol. 1, pp. 265–274. University of Magdeburg, Institute for Simulation and Graphics, Magdeburg (2009)

15. Lesieur, M., Metais, O., Comte, P.: Large-Eddy Simulations of Turbulence. Cambridge University Press, Cambridge (2005)

16. Noack, B.R., Mezić, I., Tadmor, G., Banaszuk, A.: Optimal mixing in recirculation zones. Phys. Fluids, **16**(4), 867–888 (2004)

17. Noack, B.R., Schlegel, M., Ahlborn, B., Mutschke, G., Morzyński, M., Comte, P., Tadmor, G.: A finite-time thermodynamics of unsteady fluid flows. J. Non-Equilib. Thermodynam., **33**(2),103–148 (2008)

18. Panton, R.L.: Incompressible Flow. 3rd edn, Wiley-Interscience, Hoboken, New Jersey (1996)

19. Peacock T., Dabiri, J.: Introduction to focus issue: Lagrangian coherent structures. Chaos, **20**(1), 017501 (2010)

20. Reininghaus, J., Kasten, J., Weinkauf, T., Hotz, I.: Combinatorial feature flow fields: tracking critical points in discrete scalar fields. Technical Report 11-02, Konrad-Zuse-Zentrum für Informationstechnik Berlin (2011)

21. Rom-Kedar, V., Leonard, A., Wiggins, S.: An analytical study of transport, mixing and chaos in an unsteady vortical flow. J. Fluid Mech., **214**, 347–394 (1990)

22. Salzbrunn, T., Garth, C., Scheuermann, G., Meyer, J.: Pathline predicates and unsteady flow structures. Vis. Comput., **24**(12), 1039–1051 (2008)

23. Shadden, S.C.: A dynamical systems approach to unsteady systems. PhD thesis, California Institute of Technology, Pasadena CA (2006)

24. Shadden, S.C., Lekien, F., Marsden, J.E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. Physica D, **212**(3–4), 271–304 (2005)

25. Stuart, J.T.: On finite amplitude oscillations in laminar mixing layers. J. Fluid Mech., **29**, 417–440 (1967)

26. Theisel H., Seidel, H.-P.: Feature flow fields. In: VisSym '03: Proceedings of the Symposium on Data Visualization 2003, pp. 141–148, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association (2003)

27. Tricoche, X., Wischgoll, T., Scheuermann, G., Hagen, H.: Topology tracking for the visualization of time-dependent two-dimensional flows. Comput. Graph., **26**, 249–257 (2002)

28. Weinkauf, T., Theisel, H., Van Gelder, A., Pang, A.: Stable feature flow fields. IEEE Trans. Vis. Comput. Graph., TVCG **17**(6), pp. 770–780 (2010) accepted

29. Williamson, C.H.K.: Vortex dynamics in the cylinder wake. Ann. Rev. Fluid Mech., **28**, 477–539 (1996)

30. Yule, A.J.: Investigations of eddy coherence in jet flows. In: Jimenez, J. (ed.), The Role of Coherent Structures in Modelling Turbulence and Mixing, vol. 136, Lecture Notes in Physics, Berlin Springer Verlag, pp. 188–207 (1981)

Ridge Concepts for the Visualization of Lagrangian Coherent Structures

Benjamin Schindler, Ronald Peikert, Raphael Fuchs, and Holger Theisel

1 Introduction

There have been various attempts to find lines or surfaces that characterize the structures of an unsteady flow field in a similar way as topological skeletons [11] characterize a steady flow. The most influential approach is probably Haller's definition [9] of Lagrangian coherent structures (LCS) as the *ridges* in the scalar field of finite-time Lyapunov exponents (FTLE). Numerical experiments [21] have shown that these ridges coincide well with material lines (in 2D flow) or material surfaces (in 3D flow). However, this coincidence is not something that could be formulated as a mathematical statement, because the FTLE contains a parameter, the integration time, and also because there are multiple definitions for a ridge. Such a statement would for example hold in the limit of infinite integration time if a steady flow is assumed. Then, the Lyapunov exponent would be constant per trajectory, and the ridges, under any reasonable definition, would be material lines. In scientific visualization, we have to deal with finite time domains, and here it turns out that FTLE ridges can deviate significantly from material lines or surfaces.

An obvious error metric for comparing different types of FTLE ridges, FTLE watersheds and related features is the flux through these lines or surfaces. The flux per unit length can be computed as the normal component of the velocity vector minus the velocity with which the feature moves in the normal direction. The latter can be estimated by tracking the feature over a small time interval.

B. Schindler (✉) · R. Peikert · R. Fuchs
ETH Zurich, Rämistrasse 101, 8092 Zurich, Switzerland
e-mail: bschindler@inf.ethz.ch; peikert@inf.ethz.ch; raphael@inf.ethz.ch

Holger Theisel
University of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
e-mail: theisel@isg.cs.uni-magdeburg.de

Besides comparing different types of FTLE ridges with respect to this error metric, a second goal of this paper is to explore other FTLE-related concepts. The definition of FTLE contains a *normalization* and a *natural logarithm*. These two operations are inherited from the (infinite-time) Lyapunov exponent, where they are needed to make it independent of the starting time of a trajectory and, under the conditions of the Oseledec theorem, also independent of the trajectory. In the finite-time case these independences do not hold. Therefore, neither the logarithm nor the normalization (which both are monotone functions) are needed. By omitting the two functions, the largest eigenvalue of the (right) Cauchy-Green deformation tensor is obtained. This leads us to the study of the eigenvalues and eigenvectors of this tensor as a basis for an alternative definition of LCS lines and surfaces.

2 Related Work

FTLE have been used by the fluid dynamics community for visualizing flow phenomena such as flow separation [21], vortex rings [20], transport barriers [16], or flows generated by jellyfish [18]. In the visualization community, algorithms for efficient FTLE computation and subsequent visualization techniques were developed in recent years. Garth et al. [7] presented a technique for 2D flow, and later for 3D flow [6] where direct volume rendering was used to visualize the FTLE field. As an alternative to a full computation of a 3D FTLE, they proposed to compute the FTLE only on planar cross sections for gaining efficiency. Sadlo et al. [22] addressed the efficiency issue with a hierarchical approach where the sampling grid is refined only in the vicinity of ridges. In a later paper [23], grid advection was used in order to exploit temporal coherency of time-dependent velocity data. Lipinski and Mohseni [15] present a method to track ridges over time to reduce the amount of FTLE computations for time-dependent data.

Besides algorithmic aspects, visualization researchers also found novel uses for FTLE. Bürger et al. computed an FTLE field for steering particle seeding in a particle-based visualization [1]. Also in terms of applications there is a recent diversification of uses of FTLE. They have been used for visualizing the flow in the small bronchial tubes by Soni et al. [24] and for cell aggregation by Wiebel et al. [29].

Shadden et al. [21] give an analytical and numerical analysis of the suitability of ridges of FTLE for being used as LCS. Their demonstration that FTLE ridges behave nearly as material lines is based on two non-standard ridge definitions that both have the problem that they are overdetermined (see Appendix). While they express desirable properties of ridges, the fact that practical ridges deviate from these is a motivation to study the effect of the chosen ridge definition on feature extraction from FTLE or related data.

3 Background

The *flow map* (in mathematics sometimes simply called the *flow*) of a vector field $\mathbf{u}(\mathbf{x}, t)$ is the map from the point where a massless particle is seeded at time t_0 to the point where it is located at time t. The flow map is denoted by $\Phi_{t_0}^t(\mathbf{x})$. Formally, it is described by the following initial value problem

$$\frac{\partial}{\partial t} \Phi_{t_0}^t(\mathbf{x}) = \mathbf{u}(\Phi_{t_0}^t(\mathbf{x}), t), \qquad \Phi_{t_0}^{t_0}(\mathbf{x}) = \mathbf{x}. \tag{1}$$

Its gradient $\mathbf{F} = \nabla \Phi_{t_0}^t(\mathbf{x})$ leads to the *(right) Cauchy-Green deformation tensor* $\mathbf{C} = \mathbf{F}^\top \mathbf{F}$. With this, the *finite-time Lyapunov exponent (FTLE)* can be defined as

$$\sigma(\mathbf{x}, t_0, t) = \frac{1}{|t - t_0|} \ln\left(\sqrt{\lambda_{\max}(\mathbf{C})}\right) \tag{2}$$

where $\lambda_{\max}(\mathbf{C})$ denotes the largest eigenvalue of $\mathbf{C}$.

The FTLE therefore describes the rate of separation of a pair of particles, where the maximum is taken over all spatial orientations of the pair. The FTLE can also be computed for a flow map going backward in time. In that case it describes attraction rather than repulsion.

3.1 Watersheds, Height Ridges, and Section Based Ridges

Even though ridges in a 2D height field are easily recognized by the human eye, there is not a unique ridge definition. Generally, a ridge is a lower-dimensional manifold in a scalar field that generalizes the notion of a local maximum.

A natural definition of a ridge is the *watershed*. For a 2D scalar field the watershed is obtained by integrating the gradient of the field, starting at a saddle point. Intuitively, this means to walk uphill from a pass to a peak, always taking the direction of the steepest slope. In topological terms, watersheds are part of the topological skeleton of the gradient field. This way, extension to 3D scalar fields is straightforward: They are the unstable 2D manifolds of saddle points. Sahner et al. [26] used such 2D watersheds for their concept of strain skeletons in 3D velocity fields.

Height ridges are a generalization of local maxima. A (maximum convexity) height ridge of co-dimension one is given by the points which have a zero first derivative and a negative second derivative if derivatives are taken in the direction of the minor eigenvector of the Hessian [3]. Here, the minor eigenvector refers to the eigenvector associated with the smallest *signed* eigenvalue. Often this set of points is further reduced by additional constraints in order to remove false positives [14, 19].

Section based ridges are obtained by finding local maxima in a set of parallel $n - 1$-dimensional sections and connecting them [12]. Obviously, the orientation chosen for the sections influences the result, and reasonably good results can be expected only if the ridge is roughly orthogonal to the sections. Section based ridges have the advantage that they do not require second derivatives.

In 3D velocity fields, it can be observed that the FTLE field often has surface-like maxima, defining structures which are approximately orthogonal to the major eigenvector. Therefore, a height ridge of co-dimension one can be used to represent such structures [22].

4   Tensor Field Lines

LCS are obtained by applying any of the above ridge definitions to the FTLE field. Since the FTLE field only depends on the eigenvalues of $\mathbf{C}$, an alternative is to include also the eigenvectors of $\mathbf{C}$. The background of this is: The Cauchy-Green tensor $\mathbf{C}$ is a positive symmetric tensor, and $\mathbf{U} = \mathbf{C}^{1/2}$ is the right stretch tensor in the polar decomposition $\mathbf{F} = \mathbf{RU}$, while $\mathbf{R}$ is a rotation matrix. The eigenvectors $\mathbf{N}_i$ of $\mathbf{U}$ (being also the eigenvectors of $\mathbf{C}$) are orthogonal and contain the directions of maximal and minimal stretch. The eigenvectors $\mathbf{n}_i$ of the left stretch tensor $\mathbf{V}$ (with $\mathbf{F} = \mathbf{VR}$) are the vectors $\mathbf{N}_i$ after rotation, $\mathbf{n}_i = \mathbf{RN}_i$ (see Fig. 1).

A *tensor field line* is obtained by integrating along the minor eigenvector of $\mathbf{C}$. The seed point for this integration can be a local FTLE maximum or another point which is assumed to lie on an LCS. Integration is stopped when a degenerate point is approached where the eigenvector direction becomes undefined. The resulting structures have the property that they are aligned with the direction of maximal stretching. Tensor field lines are included in this study as an alternative to ridges.

5   C-Ridges

Section based ridges are biased by the sectioning direction. However, this can be largely reduced by using sections that are taken orthogonal to the predicted ridge direction. In our case of ridges of an FTLE field $\sigma(\mathbf{x})$, such an orthogonal direction exists, namely the major eigenvector $\mathbf{N}_1$ of $\mathbf{C}$, as was explained in Sect. 4. The ridge obtained this way can be expressed by

$$\mathbf{N}_1 \times \nabla\sigma = 0 \quad \text{and} \quad (\mathbf{H}_\sigma \times \mathbf{N}_1) \times \mathbf{N}_1 < 0 \tag{3}$$

where $\mathbf{H}_\sigma$ denotes the Hessian of σ. It differs from the height ridge only in the use of the eigenvector $\mathbf{N}_1$ which for a height ridge must be replaced by the minor eigenvector of $\mathbf{H}_\sigma$. This new type of ridge, restricted to points where $\sigma > 0$, fulfills two of the conditions of Haller's new concept of *weak LCS* (cf. [10], Definition 7),
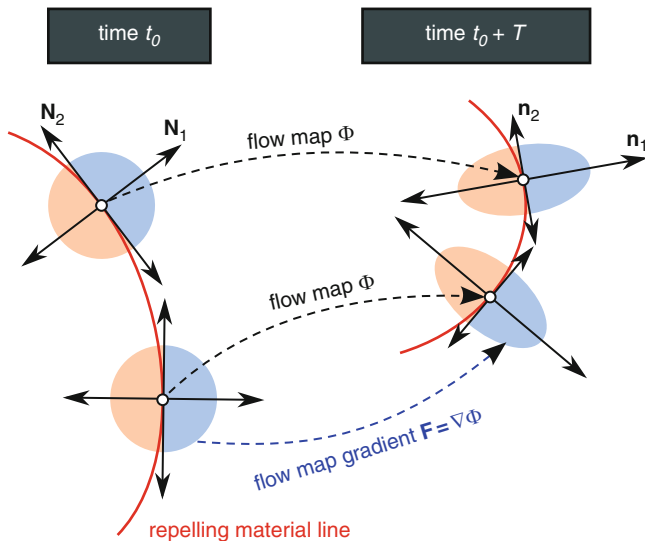
Fig. 1 Tensor field line of $\mathbf{C} = \mathbf{F}^{\top}\mathbf{F}$ acting as a repelling material line. When advected, its particles are mapped by Φ and their neighborhoods by $\mathbf{F}$

Fig. 2 Radial integration method used for $\mathbf{C}$-ridge surfaces
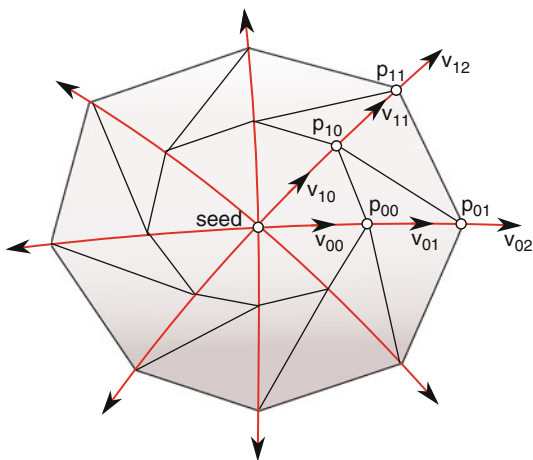
while the other two (material surface, the surface itself is orthogonal to $\mathbf{N}_1$) can hold in general only in the limit $t \to \infty$. For brevity of notation, we will use the name $\mathbf{C}$-ridges for these ridges.

The $\mathbf{C}$-ridge can be extended to a surface in 3D by taking a local FTLE maximum as the seed point and a tangential plane orthogonal to the major eigenvector. In Fig. 2 we illustrate the integration approach for the $\mathbf{C}$-ridge. At the seed point the ridge normal vector is given by the major eigenvector $\mathbf{N}_1$ of $\mathbf{C}$. We choose a vector $\mathbf{v}_{00}$ normal to $\mathbf{N}_1$ and rotate this vector around $\mathbf{N}_1$ in discrete angular steps, giving

vectors $\mathbf{v}_{0i}$. We denote the jth integration step in the ith direction as $\mathbf{v}_{ij}$ and the resulting point as $\mathbf{q}_{ij}$. At $\mathbf{q}_{ij}$ a correction step is done in the direction of the local eigenvector $\mathbf{N}_1$, maximizing the FTLE value σ. The result is denoted by $\mathbf{p}_{ij}$. The next step $\mathbf{v}_{i\,j+1}$ is then computed as the projection of the previous stepping direction $\mathbf{v}_{ij}$ onto the normal plane of $\mathbf{N}_1$ at the point $\mathbf{p}_{ij}$. Integration is stopped when σ drops below a given threshold.

It is worth clarifying that both height ridges and $\mathbf{C}$-ridges can be detected locally, i.e., they would not require a seed point. However, local maxima of FTLE always lie on a ridge, and by taking (a subset of) the maxima as the seed points a subset of "important" ridges is obtained. This approach is also computationally more efficient, even though it means that duplicates must be checked for. Furthermore, it allows ridges to be tracked over time by tracking the local maximum, which is easily done by following the steepest ascent.

As a side note, the same procedure without the correction step would give the 3D analog to the tensor field line, which has been used also for visualization [28]. However, it is well known [25] that a surface exactly orthogonal to the major eigenvector does not exist in general.

6 Error Metric for the Material Line/Surface Property

Material lines and surfaces have the property that there is zero flux crossing the line or surface. A straightforward error metric is therefore based on the flux. When computing the flux we have to take relative velocities because the line or surface is moving. Formally, for the case of a line γ in 2D, this is

$$\int_{\gamma} (\mathbf{u}(\mathbf{x}(s,t),t) - \mathbf{c}(s,t)) \times \mathbf{n}(s,t) ds \qquad (4)$$

where $\mathbf{x}(s,t)$ is the position of the curve point with parameter value s at time t, $\mathbf{u}(\cdot,t)$ the velocity field, $\mathbf{c}(\mathbf{s},\mathbf{t})$ is the point's velocity as given by its tracking, and $\mathbf{n}(s,t)$ is the curve normal.

For a local measure, the flux is taken per unit length (or unit area). Its computation requires computing LCS for a sequence of time steps. LCS curves (or surfaces) are tracked over time and the speed orthogonal to the LCS is derived from its motion. This speed is compared to the local velocity field, again projected onto the orthogonal space of the LCS. The difference is the *local flux*, i.e., the flux per unit length or area. Since the local flux is commensurate with a velocity, it can be *normalized* by dividing it by the local velocity magnitude. Normalized local fluxes were used by Shadden et al. [21]. This error metric can be used for line-type structures in 2D domains and for line-type and surface-type structures in 3D domains.

The *tracking* of an LCS over time is done by tracking its seed point. For the majority of the above feature definitions the seed point is a local FTLE maximum, which is simply tracked by following the gradient (steepest ascent method). The remaining feature types, watersheds and tensor field separatrices, are seeded at saddle points of the FTLE field and degenerate points of the tensor field **C**, respectively. These other seed points could be tracked by a critical point tracking method [8, 27]. For our study, we did a simple tracking by proximity.

However there is no one-to-one mapping between these seeds and the FTLE maxima where ridges are seeded. At the initial time step, the best matching watershed and tensor field separatrix have to be found manually. After each tracking step it is verified that the watershed still extends to the FTLE maximum and the separatrix does this sufficiently close. If not, this means that a "bifurcation" event has happened. Then the tracking is stopped, and the procedure is re-initialized.

7 Results

In this section we evaluate the differently defined LCS on series of time steps of two 2D vector fields. We also use a 3D dataset from [22] for a qualitative comparison of height ridges and the proposed **C**-ridges.

7.1 "Double Gyre" Example

The double gyre is an analytic unsteady 2D vector field which was used by Shadden et al. [21] to demonstrate that saddles of vector field topology can deviate from the point of actual saddle behavior. The field is defined by

$$u(x, y, t) = -\pi A \sin(\pi f(x, t)) \cos(\pi y)$$
$$v(x, y, t) = \pi A \cos(\pi f(x, t)) \sin(\pi y) \frac{df(x,t)}{dx} \tag{5}$$

where

$$f(x, t) = \varepsilon \sin(\omega t) x^2 + (1 - 2\varepsilon \sin(\omega t)) x \tag{6}$$

and $A = 0.1, \omega = \pi/10, \varepsilon = 0.25$. First, we use long-time FTLE to reproduce the findings from Shadden et al. [21]. Then, different ridge types are compared using short-time FTLE. This makes sense because the difference in flux is expected to be more pronounced when short integration times are used.

7.1.1 Results for Long-Time FTLE

As a first test, we measured the flux through a height ridge of an FTLE of the "double gyre" computed with integration time $T = 30$, and compared this with the findings of Shadden et al. [21]. Their Fig. 8 can be roughly confirmed with our method of computing height ridges of FTLE. Also the flux through this ridge is consistent with their values. However, our height ridge computed for the same parameter values ($A = 0.1, \omega = 2\pi/10, \varepsilon = 0.1$) passes the point $x = 1.019, y = 0.9370888$ that means that the ridges disagree by about 0.0003. We computed the FTLE field using the C version of Netlib's RKF45 with relative error setting of 10^{-9} (and consistently also with 10^{-8}), and with a grid resolutions of 2×10^{-7} and 2×10^{-6} (see Fig. 3b). We computed a flux less than 10^{-5} (five times less than computed by Shadden et al.) with either of the two resolutions. The normalized flux, as defined in Sect. 6 is less than 0.0002.

7.1.2 Results for Short-Time FTLE

The goal of [21] was to investigate the sharp FTLE ridges that result from flow maps taken over a long integration time T. It is mentioned that FTLE ridges are less suitable for short times T. However, in the practice of data visualization the time domain is bounded by the available datasets, and very often it is too short to develop sharp ridges. If ridges are "soft", the different ridge definitions lead to the extraction of significantly different curves. It is one of the main purposes of this study to
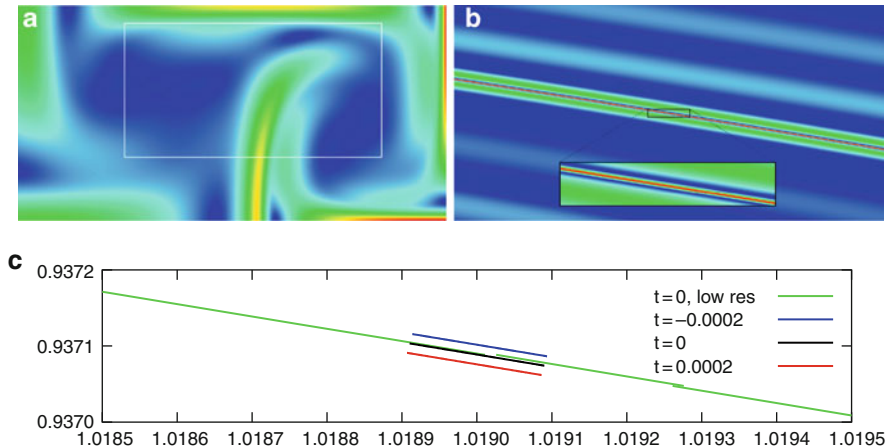
Fig. 3 Short-time and long-time FTLE of the "double gyre" vector field. (**a**) FTLE at $t = 0$, integration time $T = 3$. The white rectangle marks the close-up region used in Fig. 4. (**b**) FTLE with integration time $T = 30$ in the region discussed in [21]. (**c**) Extracted height ridge on close-up region at $t = 0$ and $t = \pm 0.0002$

compare these definitions and the resulting ridge curves in terms of their geometry and their cross flux which quantifies their deviation from being material lines.

In Fig. 4, the main ridge of a short-time ($T = 3$) FTLE of the "double gyre" is shown. The height ridge (red), section-based ridge (orange), **C**-ridge (green) and tensor field line (blue) are all seeded at the local FTLE maximum near the x-axis. The FTLE watersheds (black) are seeded at a saddle point. Since there are several watersheds converging to the same local maximum, the best one in terms of cross-flux was taken for the comparison. The separatrix of the tensor field **C** (magenta) is seeded at a degenerate point. These do not converge to FTLE maxima as they are not directly related to the FTLE field. For the comparison we took the one separatrix that approaches the chosen FTLE maximum most closely.

The set of 5x5 particles in Fig. 4 shows that all curves have some particles crossing them in the time interval $(1.0, 2.0)$. Toward the bottom of the image, all feature lines except the tensor field separatrix (magenta) coincide with the two FTLE watersheds (black). The height ridge (red) originating at the FTLE maximum below the bottom of the image curves to the right at $t=1.0$ and later connects with the branch reaching to the left. In either stage the long branch of the height ridge follows closely one of the two watersheds. Also the **C**-ridge (green) closely follows it. Some of the curves are crossed by fewer particles, such as the y-section ridge (orange), the tensor field separatrix, and in particular the tensor field line (blue) which is consistent with the flux plot in Fig. 5b for the same time interval.

The evaluation result using our error metric is shown in Fig. 5 where normalized fluxes are plotted for the various types of ridge curves. In Fig. 5a fluxes are measured at the lower end of the ridge. Since this ridge is quite pronounced here, most curves nearly coincide. The exception is the tensor field separatrix which does not run on top of the ridge. In Fig. 5b where the ridge starts to fall off, it turns out that height ridges, watersheds and **C**-ridges behave similarly throughout the time domain $t = 0, \cdots, 10$.
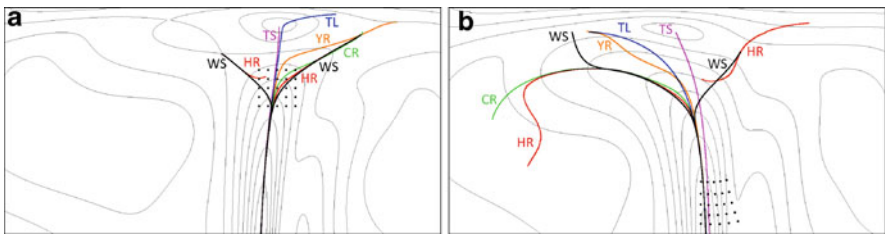
Fig. 4 Close-up of the "double gyre" with advected particles and the tracked feature lines: height ridges (HR), FTLE watersheds (WS), y-section ridge (YR), **C**-ridge (CR), tensor field line (TL), tensor field separatrix (TS). (**a**) Time $t=1.0$. (**b**) Time $t=2.0$
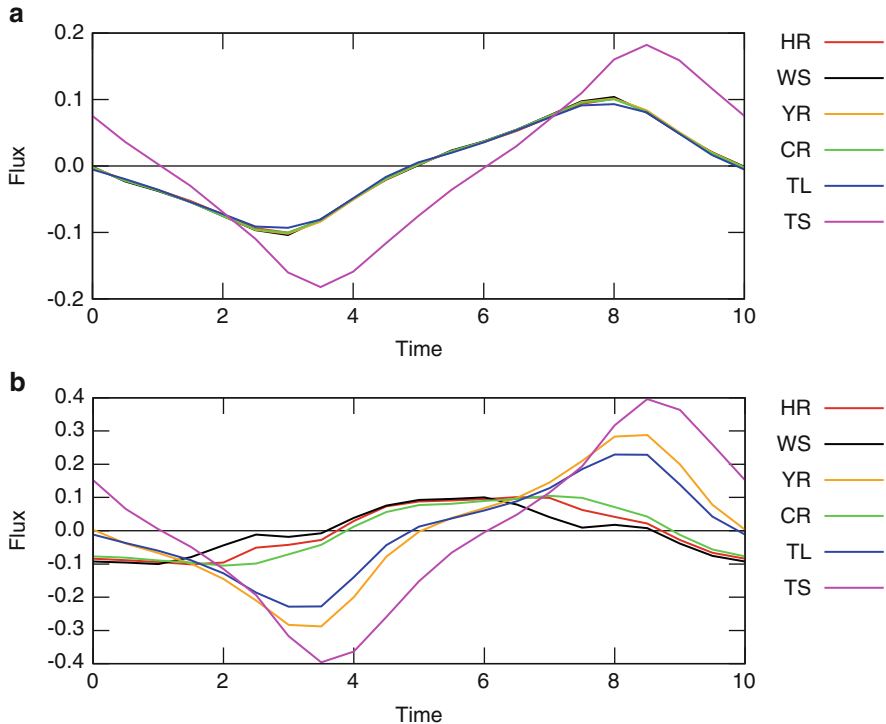
Fig. 5 Fluxes per unit length measured for the various feature lines over the full periodic time range $t = 0, \cdots, 10$ of the "double gyre" field. FTLE ridges: height ridges (HR), $y = const$ section ridges (YR), **C**-ridges (CR), watersheds (WS). Field lines of **C**: seeded at FTLE max (TL), at degenerate point (TS). Integration time used for computing C and FTLE was $T = 3$. (**a**) Fluxes through feature lines at $y = 0.2$. (Integration time $T = 3$). (**b**) Fluxes through feature lines at $y = 0.6$. (Integration time $T = 3$)

7.2 Square Cylinder Example

This vector field is a transient 3D simulation of the flow about a square cylinder (or cuboid) done with Ansys' finite-volume solver CFX. The cuboid vertically extends only to one half of the computational domain, therefore the fluid (water) can flow over the top and instead of a classical von Kármán vortex street a chain of more three-dimensionally shaped vortices appears as can be seen in Fig. 6a where vortices are visualized by a λ_2 isosurface.

We use the full 3D version of the data set for a qualitative analysis. We implemented the tensor field surface as discussed in Sect. 5. Fig. 6b shows that it is consistent with the FTLE color-mapped on the slice. For the quantitative analysis we use this 2D slice on the half height of the cuboid. The reason is that not all of the feature lines have a well-defined 3D analog. We cropped the slice to a rectangle of physical size 0.18 by 0.06. Only ridges passing through the seed point as indicated in Fig. 7 are shown. Since the features extend mostly along the x-axis, we also

added ridges based on constant x sections to the comparison. The results are plotted in Fig. 8, where we selected time steps $t = 0.13$ and $t = 0.14$ and manually picked as the seed point one of the FTLE maxima.

Similar observations as in the first example can be made here. Through all structures there is non-negligible flux. The three types of ridges give very similar results with the **C**-ridge being slightly but consistently better than the other two. The tensor field lines which geometrically deviates from the ridges has similar repulsion strength, but the cross-flux differs for some of the time steps quite significantly, e.g., at $t = 0.14$ there is almost no flux. The time steps $t = 0.13$ and $t = 0.14$ are shown in Fig. 7.

Summarizing the evaluation of both examples, we observe that both tensor field lines and FTLE watersheds can drift away at some point from an FTLE ridge. This is because these methods are based on integration and lack a local correction toward a maximum. With the proposed error metric, tensor field lines cannot be qualified as better or worse than FTLE ridges. We included ridges based on axis-aligned sections only for the purpose of comparison. Their simple definition and the fact that the FTLE structures happen to be roughly oriented in axes directions makes them useful as a reference.

Section based ridges with sections taken perpendicular to the minor eigenvector of **C** are, however, a valid alternative. Not only can **C**-ridges be computed with just first derivatives of FTLE (which is sufficient for iteratively finding local maxima) but also the directional information contained in **C** is exploited, which is missing in the FTLE field and therefore in its height ridges. In all cases where watersheds and
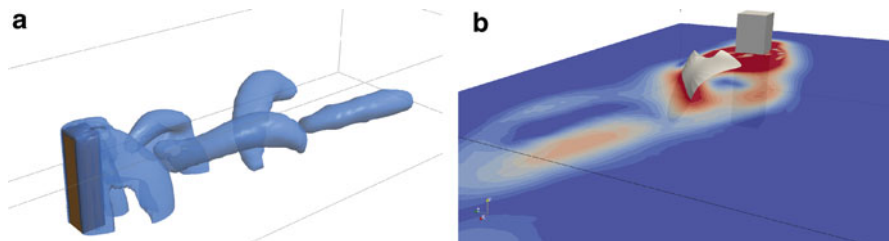
Fig. 6 Flow about a square cylinder. (**a**) Vortices (isosurfaces of λ_2). (**b**) **C**-ridge surface and FTLE in a slice at half height of the cylinder
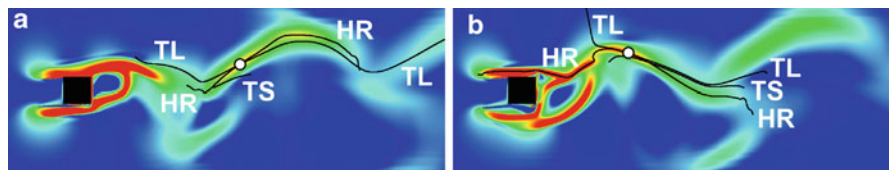
Fig. 7 Height ridge (HR) passing the selected local FTLE maximum (*circled*), tensor field line (TL) seeded at the same point, tensor field separatrix (TS) seeded at the nearest degenerate point. (**a**) Time $t = 0.13$. (**b**) Time $t = 0.14$
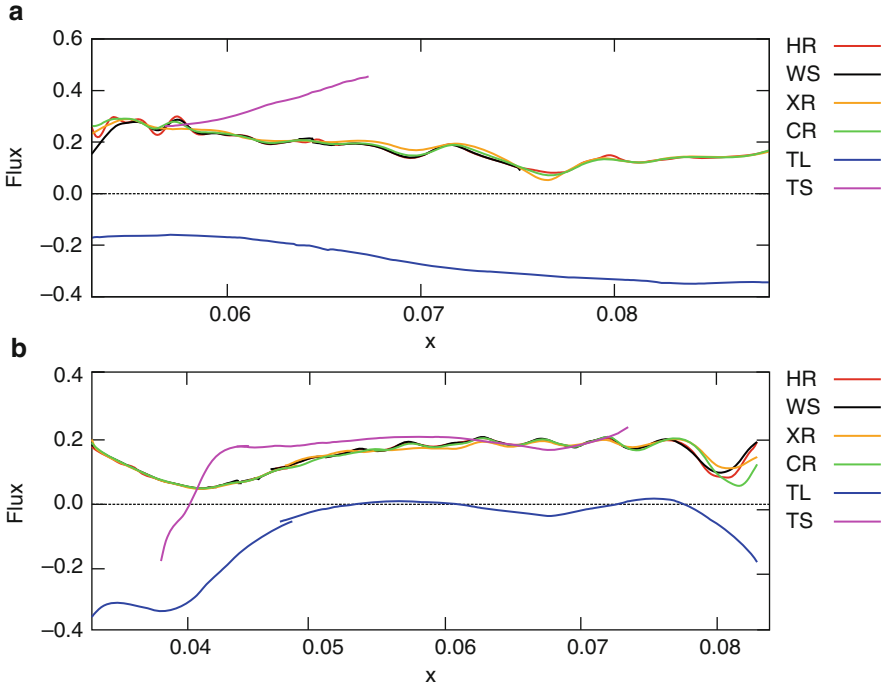
Fig. 8 Error of feature lines at selected time steps in the square cylinder example. HR: height ridge, WS: watershed, XR: $x = const$ section ridge, CR: **C**-ridge, TL: tensor field line, TS: tensor field separatrix. (**a**) Flux per unit length through feature lines, $t = 0.13$. (**b**) Flux per unit length through feature lines, $t = 0.14$

height ridges are in good accordance, the **C**-ridge was found to be also consistent with them.

7.3 Francis Turbine Example

This vector field is an unsteady CFD simulation of an entire Francis turbine. High FTLE values can be found at the lower end of the draft tube where it branches in two parts. This region of interest was already used by Sadlo et al. [22] to compute height ridges with an optimized method. Here, we compare height ridges and **C**-ridges in terms of mesh quality. By using the following brute-force method, we make sure that mesh quality is not influenced by any optimization technique.

On a regular grid the flow map Φ is computed and from this the gradient **F**, the Cauchy-Green tensor **C**, the FTLE σ as well as its gradient and Hessian $\mathbf{H}_\sigma$. All derivatives are computed by convolving with derivatives of Gaussian. Depending on the choice of height ridges or **C**-ridges, the major eigenvector of $\mathbf{H}_\sigma$ or **C** is computed on all grid nodes. Then, in a loop over the cells, the Marching
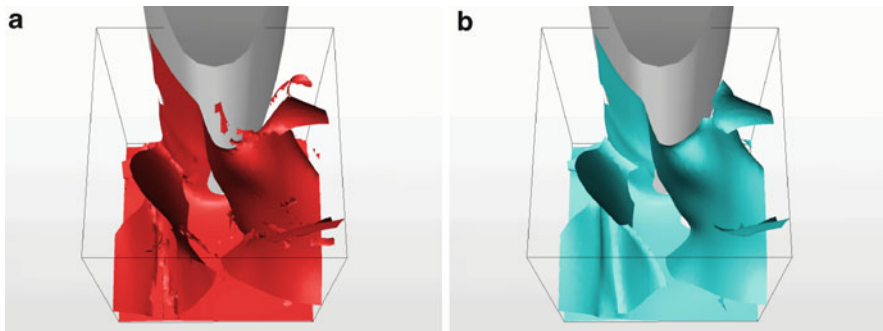
Fig. 9 Comparison of height ridge (**a**) and **C**-ridge (**b**) in Francis turbine data, both extracted with Marching Ridges method

Ridges algorithm [4] is applied. This consists of using principal component analysis to convert the eigenvectors into a consistently oriented vector field **e** and then computing the zero-level isosurface of $\mathbf{e} \times \nabla\sigma$. The Asymptotic Decider [17] method is used for the disambiguation. Finally, all triangles are trimmed at the isoline $\sigma = \sigma_{\min}$ for removing parts below a threshold and at the isoline $(\mathbf{H}_\sigma \, \mathbf{e}) \times \mathbf{e} = 0$ to remove non-ridge parts ("connector surfaces" [2]).

Fig. 9 shows that while the two types of FTLE ridges generally coincide, the height ridge is much noisier. The standard deviation of the Gaussian kernel is a parameter that affects the smoothness of the ridges. This effect of the scale on ridges can be studied in a scale-space setting [5, 13]. In our study, several fixed values were used (for Fig. 9 twice the flow map resolution), and the quality difference between the two types of ridges was found to be consistent. It can therefore be attributed to the fact that second derivatives of σ are needed in the **C**-ridge only for the trimming of triangles, but not for the ridge extraction itself.

8   Conclusion

By computing the flux through various types of ridges of FTLE fields, we confirmed the findings made in [21] and we demonstrated that for shorter integration times the normalized flux can be 0.1 or more.

We introduced the **C**-ridge and showed that it is a good alternative to FTLE height ridges, because it proved to be slightly better quite consistently. The main advantage of the **C**-ridge is that it only uses eigenvectors of **C**, i.e., information that is basically available when the FTLE field is computed, and first derivatives of FTLE for the maximum search. Height ridges require second derivatives which is numerically problematic since the FTLE field already required a numerical differentiation.

Appendix

In ref. [21] two alternative ridge definitions are used, both of which are overdetermined and can therefore not be strictly fulfilled in general. In definition 2.2 of the *second-derivative ridge*, condition SR1 says that the curve must everywhere be tangent to the gradient of the FTLE field, i.e., it must be a *slope line* of σ. Together with an initial condition this fully specifies the curve, e.g., a *watershed* of σ if a saddle point is taken as the seed point. Condition SR2 also prescribes the tangent direction of the ridge curve at each of its points, this time as the eigenvector of the Hessian of σ corresponding to the larger (signed) eigenvalue. An additional constraint is that the other eigenvalue must be negative. Again, condition SR2 specifies the curve up to a seed point. For practical ridges this means that at least one of these conditions holds only approximately. Condition SR2 can be transformed into the standard definition of the (maximum convexity) *height ridge* [3] by replacing the tangent direction of the ridge by $\nabla\sigma$. Height ridges, if they are sufficiently "sharp", fulfill SR1 in an approximate sense. A relaxed version of SR1 can be used to post-filter the set of computed height ridges, e.g., by allowing for a certain maximum angle between the ridge and $\nabla\sigma$ [19]. Definition 2.1 of the *curvature ridge* is overdetermined in the same way as definition 2.2, therefore both definitions must necessarily be relaxed to become practically usable as, for example, watersheds, height ridges or surface creases.

References

1. Bürger K., Kondratieva P., Krüger J., Westermann R.: Importance-driven particle techniques for flow visualization. In: Proceedings of IEEE VGTC Pacific Visualization Symposium 2008, Los Alamitos, pp. 71–78. IEEE, Kyoto (2008)
2. Damon J.: Properties of ridges and cores for two-dimensional images. J. Math. Imaging Vis. **10**(2), 163–174 (1999)
3. Eberly D.: Ridges in image and data analysis. In: Computational Imaging and Vision. Kluwer Academic Publishers (1996)
4. Furst J. D., Pizer S. M.: Marching ridges. In: Proceedings of the IASTED International Conference Signal and Image Processing, pp. 22–26. IASTED/ACTA Press, Honolulu (2001)
5. Fuchs R., Schindler B., Peikert R.: Scale-space approaches to FTLE ridges. In: Peikert R., Hauser H., Carr H., Fuchs R. (eds.) Topological Methods in Data Analysis and Visualization II, pp.283–296, Springer, Heidelberg (2012)
6. Garth C., Gerhardt F., Tricoche X., Hagen H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Vis. Comput. Graph. **13**(6), 1464–1471 (2007)
7. Garth C., Li G.-S., Tricoche X., Hansen C. D., Hagen H.: Visualization of coherent structures in transient 2d flows. In: Hege, C., Polthier, K., Scheuermann, G. (eds.) Topology-Based Methods in Visualization II, A K Peters, Springer, Heidelberg (2009)

8. Garth C., Tricoche X., Scheuermann G.: Tracking of vector field singularities in unstructured 3d time-dependent datasets. In: Proceedings of IEEE Visualization 2004, pp. 329–336. IEEE Computer Society Press, Austin (2004)
9. Haller G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Physica D **149**, 248–277 (2001)
10. Haller G.: A variational theory of hyperbolic Lagrangian Coherent Structures. Physica D, **240**(7) 574–598 (2010)
11. Helman J., Hesselink L.: Visualizing vector field topology in fluid flows. IEEE Comput. Graph. Appl. **11**, 36–46 (1991)
12. Johnston E., Rosenfeld A.: Digital detection of pits, peaks, ridges, and ravines. IEEE Trans. Syst. Man Cybernet. **5**, 472–480 (1975)
13. Kindlmann G.L., San Jose Estepar R., Smith S.M., Westin C.-F.: Sampling and visualizing creases with scale-space particles. IEEE Trans. Vis. Comput. Graph. **15**(6), 1415–1424 (2009)
14. Lindeberg T.: Edge detection and ridge detection with automatic scale selection. Int. J. Comput. Vision **30**(2), 117–154 (1998)
15. Lipinski D., Mohseni K.: A ridge tracking algorithm and error estimate for efficient computation of lagrangian coherent structures. Chaos **20**(1), 017504.1–017504.9 (2010)
16. Lekien F., Shadden S.C., Marsden J.E.: Lagrangian coherent structures in n-dimensional systems. J. Math. Phys. **48**(6) 065404 (2007)
17. Nielson G., Hamann B.: The asymptotic decider: resolving the ambiguity in marching cubes. In: Proceedings of IEEE Visualization '91, pp. 83–91. IEEE Computer Society Press, San Diego (1991)
18. Peng J., Dabiri J. O.: Transport of inertial particles by Lagrangian coherent structures: application to predator-prey interaction in jellyfish feeding. J. Fluid Mech. **623**, 75–84 (2009)
19. Peikert R., Sadlo F.: Height ridge computation and filtering for visualization. In: Fujishiro, I., Li, H., Ma, K.-L. (eds.) Proceedings of Pacific Vis 2008, Los Alamitos, pp. 119–126. IEEE, Kyoto, Japan (2008)
20. Shadden S. C., Dabiri J. O., Marsden J. E.: Lagrangian analysis of fluid transport in empirical vortex ring flows. Phys. Fluids **18**(4), 047105 (2006)
21. Shadden S., Lekien F., Marsden J.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. Physica D **212**, 271–304 (2005)
22. Sadlo F., Peikert R.: Efficient visualization of lagrangian coherent structures by filtered AMR ridge extraction. IEEE Trans. Vis. Comput. Graph. **13**(6), 1456–1463 (2007)
23. Sadlo F., Rigazzi A., Peikert R.: Time-dependent visualization of Lagrangian coherent structures by grid advection. In: Pascucci, V., Tricoche, X., Hagen, H., Tierny, J. (eds.) Topological Data Analysis and Visualization: Theory, Algorithms and Applications, pp. 151–166. Springer, Heidelberg (2011)
24. Soni B., Thompson D., Machiraju R.: Visualizing particle/flow structure interactions in the small bronchial tubes. IEEE Trans. Vis. Comput. Graph. **14**(6), 1412–1427 (2008)
25. Schultz T., Theisel H., Seidel H.-P.: Crease surfaces: from theory to extraction and application to diffusion tensor MRI. IEEE Trans. Vis. Comput. Graph. **16**, 109–119 (2010)
26. Sahner J., Weinkauf T., Teuber N., Hege H.-C.: Vortex and strain skeletons in Eulerian and Lagrangian frames. IEEE Trans. Vis. Comput. Graph. **13**(5), 980–990 (Sep 2007)
27. Tricoche X., Wischgoll T., Scheuermann G., Hagen H.: Topology tracking for the visualization of time-dependent two-dimensional flows. Comput. Graph. **26**(2), 249–257 (2002)
28. Vilanova A., Berenschot G., van Pul C.: DTI visualization with stream surfaces and evenly-spaced volume seeding. In: Deussen O., Hansen C., Keim D., Saupe D. (eds.) Proceedings of Joint Eurographics - IEEE TCVG Symposium on Visualization, Geneva, Switzerland, pp. 173–182. Eurographics Association, Konstanz (2004)
29. Wiebel A., Chan R., Wolf C., Robitzki A., Stevens A., Scheuermann G.: Topological flow structures in a mathematical model for rotation-mediated cell aggregation. In: Pascucci, V., Tricoche, X., Hagen, H., Tierny, J. (eds.) Topological Data Analysis and Visualization: Theory, Algorithms and Applications, pp. 193–204. Springer, Heidelberg (2011)

Filtering of FTLE for Visualizing Spatial Separation in Unsteady 3D Flow

Armin Pobitzer, Ronald Peikert, Raphael Fuchs, Holger Theisel, and Helwig Hauser

1   Introduction

The concept of flow plays a central role in many fields. Classical application fields are the automotive and aviation industries. The visualization of data gained from the simulation or measurement of flow processes is relevant for the domain users, as visualization has the potential to ease the understanding of complex flow phenomena.

For a good overall understanding of the flow, the identification of areas with coherent flow behavior has proved to be useful. For steady flow, methods based on *vector field topology* (VFT), as introduced to the visualization community by Helmann and Hesselink [17], provide an expressive segmentation of the flow. In the case of unsteady flow, a comparable theory is not readily available, even though a number of promising approaches and methods have been worked out in the past years. We refer to a related state of the art report [25] for an overview of topology-based methods for the visualization of unsteady flow.

One of the promising directions leading to a semantic segmentation of unsteady flow, are so-called *Lagrangian* methods. These methods focus on the motion of massless particles in the flow. The most prominent methods are related to *finite-*

A. Pobitzer (✉) · H. Hauser
Department of informatics, University of Bergen, Postboks 7803, 5020 Bergen, Norway
e-mail: armin.pobitzer@uib.no; helwig.hauser@uib.no

R. Peikert · R. Fuchs
Scientific Visualization Group, ETH Zurich, Rämistrasse 101, 8092 Zurich, Switzerland
e-mail: peikert@inf.ethz.ch; raphael@inf.ethz.ch

H. Theisel
Department of Simutation and Graphics, University of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
e-mail: theisel@isg.cs.uni-magdeburg.de
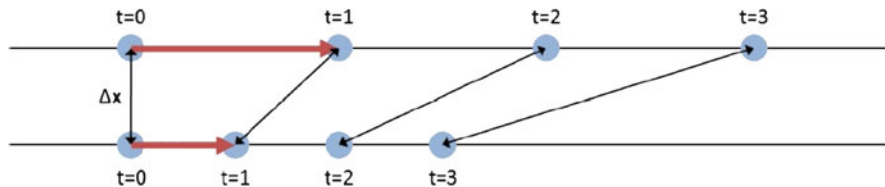
Fig. 1 Two particles traveling along straight parallel lines at different speeds: we see that the particles separate in the direction of the movement, but their paths are at a fixed distance, and will hence traverse the same regions

time Lyapunov exponents (FTLE). Haller [13, 14] shows the relation of FTLE to *Lagrangian coherent structures* (LCS) and its application to flow data.

Roughly speaking, the (maximum) FTLE gives the maximum separation rate for nearby particles over a certain time-period. When interpreting separation structures extracted from the FTLE field, such as ridges, this concept of separation, has to be kept in mind: Apart from the separation due to differences in flow directions, FTLE will also detect separation due to differences in flow magnitude. We illustrate this with a simple thought experiment:

We consider two particles that travel on straight parallel lines with constant velocity, but the one velocity being larger the other. At a certain time, these particles have a certain distance from each other. The distance between the particles increases monotonically (due to the different particle velocities), but their paths remain nonetheless parallel, leading the particles into the same area (but at different times). Figure 1 illustrates this situation.

This causes, for example, that a shear layer is a region with high FTLE values. More generally, regions of particles with parallel paths but different speeds will show this behavior. A separation concept that is not sensitive to such differences in speed would therefore define particles as "staying close" if their paths stay nearby. This concept of vicinity is called *Poincaré* or *orbital stability*. Formally, a path line is Poincaré stable if for any given $\varepsilon > 0$, there is a $\delta > 0$ such that a particle with starting distance δ to the path line stays in the ε-tube around it [19]. Although well known in theory, the definition of Poincaré stability does not provide an intuitive quantification of distance (since it would require to compare every single point on one path to all points on the other path).

From the above mentioned thought experiment we infer that separation resulting from differences in the velocity magnitude, occurs along the lines, i.e., in direction of the flow vector, while separations due to differences in flow direction will occur at an angle to the flow direction. The analysis of the deformation gradient tensor builds on the assumption of a linear mapping between the difference of the particle positions before and after advection by the flow and assumes hence that the distance between particles is locally describable by straight lines [22]. Hence, our considerations are valid for arbitrary path lines, as long as the general assumptions for the FTLE analysis are fulfilled.

The direction of the main separation can be found by analyzing the *gradient of the flow map* (in a more general setting referred to as the *deformation gradient tensor* [22]). For this purpose we use *singular value decomposition* (SVD). We show in Sect. 3 that our approach is directly derived from the geometric approach to FTLE as provided by Haller [13]. The examination of the angle between this main separation direction and the direction of the path line gives us a measure for the spatial separation that is represented by the respective FTLE values. Filtering the FTLE field with this measure then yields the separation structures representing a separation inspired by Poincaré stability. One needs to be aware of this different stability – and hence, separation – concept, and assess its meaningfulness in the case under investigation.

Accordingly, the main contribution of this paper is a new filter, to be used as a filtering step after the computation of FTLE values in unsteady flow fields, that allows to focus on those regions within the flow that lead to spatial separation.

The remainder of this paper is structured as follows: First we discuss related work. Then we introduce our proposed filtering approach, deriving it from the known theory. In the subsequent section we present results from analyzing several flow cases, applying our filtering to four simple analytical examples, the well-known "double gyre" example by Shadden et al. [29], and a CFD data set demonstrating what results we can achieve. We then discuss computational aspects of the estimation of the deformation gradient tensor and the extraction of the main separation direction. Finally we discuss results and point out future work.

2   Related Work

The visualization of flow is an active research field. Topological methods were first introduced to the scientific visualization community by Helman and Hesselink [17, 18] for both 2D and 3D steady flow fields, under the notion of *VFT*. Globus et al. [10] showed the practical relevance of VFT for computational fluid dynamics data. For a detailed survey of VFT for two and three dimensions we refer to Asimov's tutorial [1].

From the theoretical point of view, the applicability of VFT for unsteady flow has been questioned, among others, by Perry and Chong [24]. They conclude that classical VFT is only applicable to nearly steady fields. Later Shadden [29] and Wiebel et al. [33] showed this by specific examples. Very recently, Fuchs et al. [6] proposed an extended critical point concept which allows them to apply VFT in the case of unsteady flow.

Theisel et al. [31] introduce flow topology based on path lines. Path lines are the paths of massless particles that are advected by the flow. Therefore, they are inherently well suited to gain an understanding of unsteady flow.

The seminal paper of Haller [13] introduces FTLE to the analysis of flow fields. The concept of *LCS* is discussed and its connection to FTLE is revealed. LCS are – to a certain degree – the unsteady analogon of separatrices in VFT. In a follow-up

paper [14], Haller showed that LCS correspond to ridges of the FTLE field. Sadlo et al. [27] and Shi et al. [30] compare LCS to VFT and conclude that the information conveyed by FTLE is only partial as compared to VFT, missing out, for example, on vortices.

The standard algorithm for the computation of the FTLE field involves the seeding of a large number of particles in the flow and the calculation of their path lines (flow map). This is computationally challenging since it requires a high precision integration for every particle. Sadlo and Peikert [26] use adaptive mesh refinement in their ridge extraction to avoid unnecessary evaluations of the flow map. As shown by Shadden [29], LCS are "nearly" material lines. This can be exploited to speed up the algorithm. Sadlo et al. [28] present a method to extract LCS using grid advection, exploiting the temporal coherency of LCS. Lipinski and Mohseni [21] present a ridge tracking algorithm for FTLE fields that uses both temporal and spatial coherency of LCS, and give an error estimator for the difference between the advected ridge the and actual LCS. Both approaches give great speed-up compared to the standard algorithm.

As the computation of ridges usually involves the computation of higher-order derivatives, the computation will be sensitive to noise. Furthermore, some types of solvers used to simulate the flow, e.g., *spectral element methods* [32], may introduce discontinuities in higher-order derivatives.

Garth et al. [7] avoid the computation of ridges using a volume rendering approach. The authors show also that 3D FTLE might be approximated by 2D FTLE in selected cross-sections. Furthermore, the authors present an efficient approximation to FTLE fields.

Kasten et al. [20] introduce the notion of *localized FTLE* (L-FTLE). The main idea of this approach is to exchange the deformation gradient tensor with a matrix that accumulates the separation behavior along a path line. Haller and Sapsis [15] show that also the smallest FTLE is related to LCS, and can be used to compute the attracting LCS from forward standard FTLE (and vice versa). This makes computing both forward and backward FTLE obsolete and, hence, saves costly computations.

To the best of our knowledge, no attempts have been made yet to use the directional information inherent to the definition of FTLE in visualization. Obermaier et al. [23] use iteratively deformed ellipsoids to visualize volume deformation along trajectories. The deformation in every iteration step is analyzed using *SVD*, which also our approach makes use of. It is worthwhile noticing that their approach imposes divergence-freeness.

3   The Filtering Scheme

In the following, we show how the main separation direction can be computed from the directional information that is inherent to the definition of FTLE and how it can be easily derived from it.

3.1 Definition of FTLE and Its Geometric Interpretation

The concept of *finite-time Lyaponov exponents* (FTLE) is an adaptation of the concept of the classical Lyapunov exponents to the situation of a vector field which is defined over finite time only. Those fields are of practical relevance since both simulations and measurements of unsteady flow will typically yield this type of fields. Roughly speaking, the FTLE is the maximum deformation of a small neighborhood advected by the flow over a certain time-interval. This maximum deformation can be computed from the maximum eigenvalue of the (right) Cauchy-Green tensor [13, 22].

In the original paper [13], Haller gives an alternative, geometric reasoning to motivate the interpretation of the FTLE field, which yields the same formula as the standard formulation. We will use this reasoning as a starting point for our own considerations: Let $\mathbf{v}$ be a time-dependent vector field and

$$\varphi_{t_0}^{T}(\mathbf{x}_0) = \mathbf{x}(T) \tag{1}$$

the solution of the initial value problem

$$\dot{\mathbf{x}}(t) = \mathbf{v}(\mathbf{x}(t), t) \qquad \mathbf{x}(t_0) = \mathbf{x_0} \tag{2}$$

evaluated at $t = T$. $\varphi_{t_0}^{T}$ is called the *flow map*. Hence, the difference in position between two particles that are seeded at a small distance $\delta\mathbf{x}$ at time t_0 at time $t = T$ is given by

$$\varphi_{t_0}^{T}(\mathbf{x}_0 + \delta\mathbf{x}) - \varphi_{t_0}^{T}(\mathbf{x}_0). \tag{3}$$

Now, we apply a Taylor series expansion and get

$$\varphi_{t_0}^{T}(\mathbf{x}_0 + \delta\mathbf{x}) - \varphi_{t_0}^{T}(\mathbf{x}_0) = \varphi_{t_0}^{T}(\mathbf{x}_0) + \nabla\varphi_{t_0}^{T}(\mathbf{x}_0)\delta\mathbf{x} + R_1 - \varphi_{t_0}^{T}(\mathbf{x}_0) \tag{4}$$

with R_1 being an error term with $||R_1|| \in \mathcal{O}(||\delta\mathbf{x}||^2)$. Hence, in a small sphere around $\mathbf{x}_0$ we have the following approximation

$$\varphi_{t_0}^{T}(\mathbf{x}_0 + \delta\mathbf{x}) - \varphi_{t_0}^{T}(\mathbf{x}_0) \approx \nabla\varphi_{t_0}^{T}(\mathbf{x}_0)\delta\mathbf{x}. \tag{5}$$

The gradient of the flow map $\nabla\varphi_{t_0}^{T}(\mathbf{x}_0)$ is a linear operator. The maximal stretching of a δ-sphere around $\mathbf{x}_0$ is therefore

$$\max_{||\delta\mathbf{x}||\leq\delta} \left(\frac{||\nabla\varphi_{t_0}^{T}(\mathbf{x}_0)\delta\mathbf{x}||}{||\delta\mathbf{x}||} \right) = \max_{||\delta\mathbf{x}||=1} \left(||\nabla\varphi_{t_0}^{T}(\mathbf{x}_0)\delta\mathbf{x}|| \right) = ||\nabla\varphi_{t_0}^{T}(\mathbf{x}_0)||_{op} \tag{6}$$

$|| \cdot ||_{op}$ being the operator norm with respect to the usual Euclidian norm [11]. Assuming exponential growth and scaling by the integration length we get

$$\text{FTLE}(\mathbf{x}_0) = \frac{1}{|T - t_0|} \ln \left(||\nabla \varphi_{t_0}^T (\mathbf{x}_0)||_{op} \right) \qquad (7)$$

The equivalence of this formulation to the standard formula found in most papers is easy to check using basic properties of the operator norm [11, 13].

We see that the impact of the gradient of the flow map tensor on the unit sphere is the crucial aspect in the analysis of local separation using FTLE. The *SVD* is a useful tool to examine this action on the unit sphere. It is well known that a linear mapping transforms the unit sphere into an ellipsoid. The SVD gives us the opportunity to compute the main axes of this ellipsoid explicitly. More generally, the SVD of any linear mapping A is its unique representation as

$$A = U \times \text{diag}_0 (\sigma_1, \dots, \sigma_r, 0, \dots, 0) \times V^* \qquad (8)$$

where U and V are orthogonal matrices, r the rank of the matrix A, and diag_0 a block-diagonal matrix [11]. $(\cdot)^*$ denotes the transposition operator. In addition, the relation $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ holds. The columns of the matrix U are in the direction of the axes of the ellipsoid which the unit sphere is mapped to. The values σ_i are the lengths of its main axes and $\sigma_1 = ||A||_{op}$. Figure 2a illustrates this for the linear map given by $\frac{1}{4} \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$. We see that using the SVD to gain directional information about the local separation is a straight-forward extension of the original considerations of Haller.

It is worthwhile noticing that an eigenvalue decomposition of the Cauchy-Green tensor used in the standard presentation of FTLE will yield the columns of V, and not U. Unless the deformation is rotation free, these vectors will not coincide. However, the columns in V are mapped onto the columns of U. These two different sets of axes are known as the *principal spatial* and *principal material strains*,
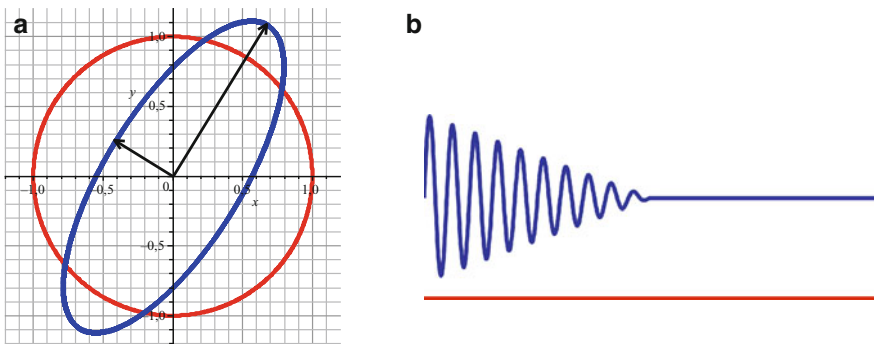
Fig. 2 (**a**) Illustration of the geometric interpretation of singular values and left-singular vectors of a linear map: The unit circle (*red*) and its image (*blue*) under a linear map. The black arrow correspond to the left-singular vectors of the map, scaled by the respective singular values. (**b**) The figure shows the trajectories of two particles moving from left two right. Although local velocities are very different we perceive them as having the same overall direction

respectively. The principal material strains provide the information on the shape of the ellipse resulting from the advection of the unit sphere $\mathscr{S}^2$ by the flow. Therefore, the use of the principal material strains to gain the directional information on the FTLE field is a straight-forward extension of the geometric approach to FTLE provided by Haller in its original paper [13]. For a thorough discussion of straining we refer to Mase [22] and Hayes [16]. Given the path line γ started in $\mathbf{x}_0$ at t_0 and integrated to $t = T$, the direction of the path line at any instant t is given by $\dot{\gamma}(t) = \mathbf{v}(\gamma(t), t)$ and the corresponding separation direction $U_{-1}(t)$ (i.e., the first column of U) is computed from $\nabla \varphi_{t_0}^t(\mathbf{x}_0)$. Hence, we can use

$$\frac{1}{T - t_0} \int_{t_0}^{T} \left| \left\langle U_{-1}(t), \frac{\mathbf{v}(\gamma(t), t)}{||\mathbf{v}(\gamma(t), t)||} \right\rangle \right| dt \tag{9}$$

as a measure for the directional difference between separation and path line starting in $(\mathbf{x}_0, t_0)$. Notice that perfect alignment of the separation direction and the flow direction, i.e., the situation we want to filter out, will cause the integrand to be 1. The absolute value has to be used since SVD may invert the orientation.

It is important to point out that this separation measure is not Galilean invariant, since it depends on properties such as velocity that are themselves not Galilean invariant. The separation measure will therefore detect path lines, that are locally parallel in the chosen frame of reference. Although Galilean invariance is an important property in general flow analysis, many interesting situations with fixed frame of reference exist, e.g., fluid flow in a tube or air flow inside of a room. Besides this, we also give an example of a separation situation below, where a Galilean invariant separation measure would fail to detect a separation that can easily be deduced from the visual inspection of the path lines in the flow (see Sect. 4.1).

In practical computations, (9) needs to be discretized. We now assume to have N samples of the path line $(\gamma(t_n))_{n=1}^{N}$. Since the velocities could change rapidly in direction, without actually affecting the perceived overall direction of the path line much, $\gamma(t_{i+1}) - \gamma(t_i)$ can be used instead of $\mathbf{v}(\gamma(t_{i+1}), t_{i+1})$ to robustify the measure. But even with this robustification, the local position differences can deviate substantially from the perceived overall direction, as we can see from Fig. 2b. We therefore choose $\gamma(t_N) - \gamma(t_i)$ instead. As the approximation to the velocities, this expression is less sensitive to fluctuation in the velocity along the path line. In addition, it uses our knowledge of where the particle will end up. In this way we estimate the overall direction of the remaining trajectory. As a convenient side effect, this estimation is also less sensitive with respect to the chosen sampling of the path line (see Sect. 5). With these considerations in mind, the discrete version of our measure for spatial separation is:

$$1 - \frac{1}{N - 1} \sum_{i=1}^{N-1} \left| \left\langle U_{-1}(t_i), \frac{\gamma(t_N) - \gamma(t_i)}{||\gamma(t_N) - \gamma(t_i)||} \right\rangle \right| \tag{10}$$

The main separation direction is the left-singular vector associated with the maximum singular value. The maximum singular value of the deformation gradient

tensor (or, equivalently, the maximum eigenvalue of the Cauchy-Green tensor) is, however, not unique by definition. In fact, all singular values σ_i might be the same, or almost the same. In addition, numerical errors may cause the two largest singular values to be of the same order. In the original definition of FTLE this does not create any problems since we are interested in the maximum only. In contrast, when looking at the angle between the associated left-singular vector and the flow vector, this situation needs special consideration. From the SVD we know that those vectors are orthogonal to each other. Hence, even if one of the vectors is almost parallel to the flow, there is a direction of comparable distortion that is nearly orthogonal to the flow. Therefore, we shall consider these points as if the main separation occurs at a large angle to the flow direction. The consideration of the third singular value is not necessary since its left-singular vector lies in the same plane orthogonal to the first left-singular vector as the left-singular vector associated with the second singular value. To account for this, we introduce a scaling factor $1 - \frac{\sigma_2(t_i)}{\sigma_1(t_i)}$ for the single summands in (10), and our final definition of the separation measure sep becomes

$$sep(\mathbf{x}_0) := 1 - \frac{1}{N-1} \sum_{i=1}^{N-1} \left(1 - \frac{\sigma_2(t_i)}{\sigma_1(t_i)} \right) \left| \left\langle U_{-1}(t_i), \frac{\gamma(t_N) - \gamma(t_i)}{||\gamma(t_N) - \gamma(t_i)||} \right\rangle \right| \quad (11)$$

Obermaier et al. [23] use the quotient of the smallest and the largest singular value to measure the overall deformation of an advection. In two dimensions, this measure coincides with the quotient in our scaling factor, the interpretation is however slightly different, as the afore reasoning shows.

3.2 The Basic Concept of the Filtering

Bringing all this together, the proposed filter scheme can be set up by four computational steps:

1. Computation of the deformation gradient tensor: This step is generally necessary in all FTLE-related algorithms and involves the integration of path lines. We save the particle positions at some intermediate time instances as well in order to compute the spatial separation. Further details are discussed in Sect. 5.
2. Computation of the SVD of the deformation gradient tensor: This step leads both to the FTLE field and the main separation directions.
3. Computation of the spatial separation of the flow using (11).
4. Focusing on regions of large angles: This focusing can be achieved by thresholding or by smooth brushing [5].

In a final step the filter is applied to the regions with high FTLE values. The above described four steps comprise the main idea for our filtering approach.

3.3 The Filter

The actual filter is then constructed by applying (smooth) brushing to the field *sep*. This brush maps values of the separation measure *sep* to the interval $[0, 1]$ and describes the degree of being *in focus*. This then corresponds to accordingly modulated opacity values in the 3D view (cf. Doleisch and Hauser [5] for further details). Hence, we can formulate our filter as $filter = brush(sep)$ Eventually, this filter is then applied to the FLTE values. This focusing is done by smooth brushing as well. The overall feature characterization function f_{sep} with range $[0, 1]$ (1 or near 1 for all locations in the flow which are considered to be part of the searched separation structure), is therefore described by

$$f_{\text{sep}} = \text{brush(FTLE)} \times \text{filter} = \text{brush(FTLE)} \times \text{brush(sep)} \tag{12}$$

The function *sep* can also be thought of as a degree of "featureness" for the feature "spatial separation," or, as *degree of interest* (DOI), using another terminology [5].

4 Case Studies

In the following we present results from the extraction of separation structures from different data sets. We demonstrate how our filtering scheme helps to focus on regions which actually separate flow compartments that move into different regions of the flow.

4.1 Synthetic Test Data

First we investigate four small analytic examples where the separation behavior can be deduced directly from the equations

$$\mathbf{v}_1(x, y) = (y, 0)^T, \qquad\qquad \mathbf{v}_2(x, y) = (y, 1)^T \tag{13}$$

$$\mathbf{v}_3(x, y) = (x - 1, 1)^T, \qquad\qquad \mathbf{v}_4(x, y, t) = (x - t, 1)^T \tag{14}$$

Notice that the field $\mathbf{v}_2$ arises from field $\mathbf{v}_1$ under the Galilean transformation $(x, y, t) \mapsto (x, y + t, t)$. The field $\mathbf{v}_4$, in turn, arises from $\mathbf{v}_3$ using the Galilean transformation $(x, y, t) \mapsto (x+t, y, t)$. Hence, it is easy to deduce from the fields $\mathbf{v}_1$ and $\mathbf{v}_3$ that the FTLE field is constant for all four fields. We investigate all four fields on the upper half plane (i.e., $y \geq 0$) and choose $t_0 = 0$ and $T = 1$. All computations for this example have been carried out using the MAPLE software package. The flow map was computed using MAPLEs seventh-eighth order continuous Runge-Kutta method `dverk78`. For estimation of the deformation gradient tensor we used
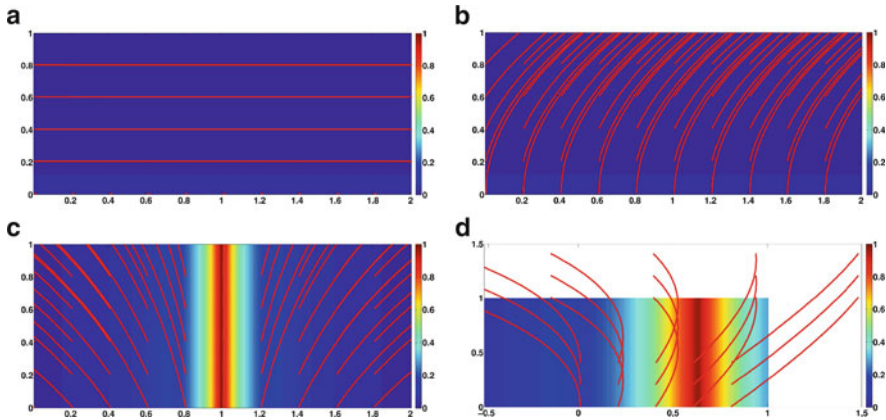
Fig. 3 Stream, resp. path, lines with the separation value as color field in the background. The left column shows the original fields, the right a Galilean transformed field. For the equations we refer to Sect. 4.1. The FTLE field is constant for all four cases. In (**a**) and (**b**) we infer from the stream lines that no spatial separation is present and the separation values are as expected close to zero (range $[0, 0.02]$). In (**c**) and (**d**) the trajectories show clear spatial separation and again the separation values coincide with the visually detected separation lines

central finite differences in the coordinate directions with spacing $h = 0.01$. For the first two fields, our separation measure *sep* is in the range $[0, 0.02]$. Hence, we expect no spatial separation. Plotting the respective stream lines of the fields shows that our filter handles both straight parallel lines (as described in the thought experiment in the introduction) as well as "locally parallel" trajectories. In contrast to the first two fields, we expect to see a clear spatial separation in the remaining two. In the first field this separation line is clearly $x = 1$, in the second field the separation line will be located right of the y-axis. Its location depends on the integration time and the speed of the observer, since this determines if particles starting on the right side of the y-axis have "enough time to turn." Our separation measure shows the expected behavior and stream, respectively path, lines plotted as a verification show the expected behavior at the separation line (see Fig. 3). The field $\mathbf{v}_4$ is an example where a Galilean invariant measure for separation would not give a response: fixing the integration time the observer speed determines where the separation line is located, and it is easy to see that any parallel to the y-axis can be achieved. Since the response would have to be the same for all observer speeds, the field would have to be constant.

4.2 Double Gyre

We demonstrate our approach in context of a well-known analytic two-dimensional example, known as the "double gyre." This has been used by Shadden et al. to demonstrate the non-usability of VFT for time-dependent flow [29], amongst others.
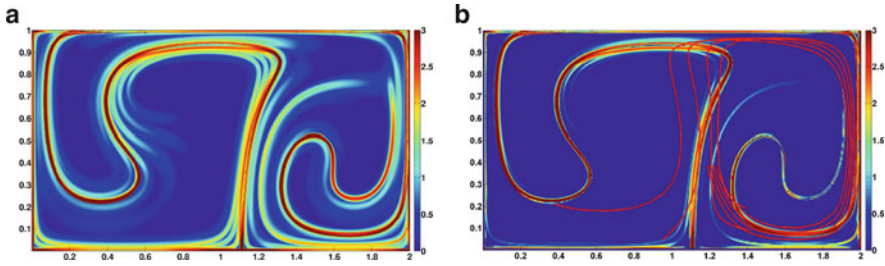
Fig. 4 The FTLE field of the double gyre with parameters $t_0 = 0$ and $T = 15$ (i.e., 1.5 periods). (**a**) The unfiltered field. (**b**) The filtered by setting FTLE values to 0 for $sep(\mathbf{x}) \leq 0.5$. Path lines confirm that the persistent ridge is indeed due to spatial separation

For the analytic definition of the field, we refer to the original paper by Shadden et al. [29]. Using the same notation as in their paper, our parameter set is $A = 1/10$, $\omega = \pi/5$ and $\varepsilon = 1/4$. The field is defined on $[0, 2] \times [0, 1] \times \mathbb{R}$. All computations for this example have been carried out using the MAPLE software package. The flow map was computed using MAPLEs seventh-eighth order continuous Runge-Kutta method `dverk78`, for estimation of the deformation gradient tensor we used central finite differences in the coordinate directions with spacing $h = 0.01$. Figure 4 shows the FTLE field with parameters $t_0 = 0$ and $T = 15$, i.e., 1.5 periods. The filtering is emulated by setting the FTLE value of points with $sep(\mathbf{x}) \leq 0.5$ to 0. We see that the filtering produces sharper ridges as the original FTLE field, highlighting in particular one ridge associated with rather low FTLE values. Seeding path lines at both sides of the ridge shows that the highlighted ridge is due to the desired type of separation, indeed.

4.3   A Bursting Dam

We apply our approach to the simulation of a bursting dam with a box-shaped obstacle. The data set consists of 48 time steps, covering the time span $[2, 120]$ (seconds) non-uniformly. The burst occurs in the first time step. We expect a recirculation zone in front (upstream) of the obstacle due to particles hitting the wall and recirculating and others getting deviated to the left and right of the obstacle. Furthermore we expect reflux on the backside of the obstacle due to pressure differences, causing particles from the end of the box to be sucked toward the obstacle, some of them ending up in front, some getting incorporated by the main flow. Right behind the obstacle we expect to see recirculation. A schematic overview of the flow can be found in Fig. 5a. The SimVis framework [4] was used for this example. We calculated the FTLE field for $t_0 = 62$ and $T = 68$, using the optimal 4th order Runge-Kutta method (sometimes referred to as the "3/8-rule"). For details we refer to Hairer et al. [12]. The usage of a even higher order integration method (which is not standard) was purely due to the fact the MAPLE software package
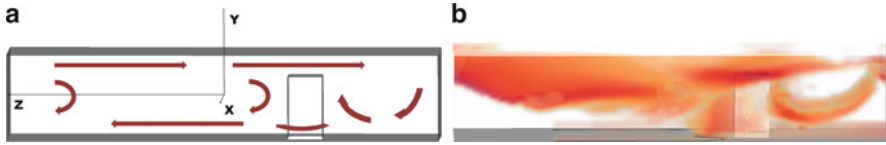
Fig. 5 (**a**) Schematic overview over the flow domain, z being the streamwise direction. (**b**) The FTLE field of a simulation of a bursting dam with parameters $t_0 = 62$ and $T = 68$. The FTLE values greater than 0.25 are brushed (smooth lower bound 0.2) . We see that we can identify expected structures around the obstacle. The upper rear part of the flow domain shows large regions with high FTLE values, presumably induced by shearing
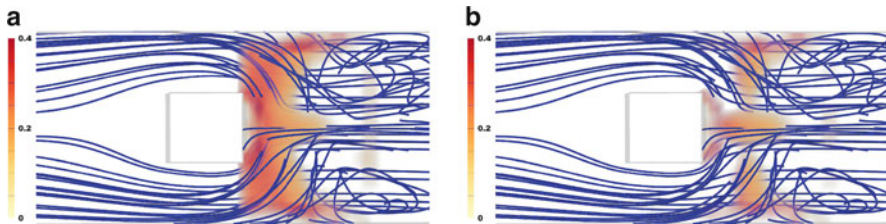
Fig. 6 The (**a**) FTLE and (**b**) filtered FTLE field upstream (right in the figures) of the obstacle in top view. The FTLE values greater than 0.25 are brushed (smooth lower bound 0.2), $sep(\mathbf{x}) < 0.45$ (smooth lower bound 0.4) is used for the filtering in (**b**). We see that the spatial separation structure stemming from particles passing on different sides of the obstacle is not clearly discernible in the unfiltered field. While adjusting the brush would not give the desired structure either, our filtering does

readily provides this method. The integration time was found empirically with the aim that not more that 15 percent of the particles seeded leave the flow domain before the end of the integration time. Figure 5b shows an overview over the FTLE field. We filter the field brushing all points with a sep-value greater or equal 0.45 (smooth lower bound 0.4). We will now investigate two regions in the flow domain more closely: The region stream-wise in front of the obstacle and the upper rear region.

4.4 In Front of the Obstacle

In front of the obstacle, we expect to detect a separation structure upstream, due to particles passing on different sides of it. We see that (Fig. 6a) this expected separation structure is not detectable from the original field. Applying our separation filter allows us to focus on this spatial separation, even though the corresponding FTLE values do not show up prominently in the original field. We added path lines to both figures to confirm that the intuitively expected separation structure indeed exists and coincides with the structure found by the filtering with our separation measure.
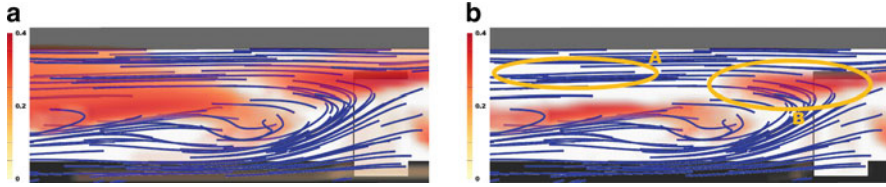
Fig. 7 A cross section of the (**a**) FTLE and (**b**) filtered FTLE field. The filtering used is the same as in Fig. 6. The ellipse (A) shows a region where the filter has a strong impact. We see that the path lines are locally parallel and show little to no spatial separation. In contrast, we see that the structure below the ellipse separates path lines moving from the left to the right (above) from those moving in the opposite direction (below). In the same fashion, the ellipse (B) indicates a structure that separates particles coming from the left and passing over the obstacle, from those moving back to the left end of the flow domain. This structure is persistent under our proposed filter

4.5 The Upper Rear Region

In the overview in Fig. 5b we see a large region with high FTLE in the upper rear part of the flow domain. Applying our filtering reduces the region to a surface separating particles moving from the back to the front (upper part) from those leaving the flow domain (lower part). We seeded particles in a cross-section in order to validate the result from the filtering. We see that the particles in the region delineated by the ellipse (A) in Fig. 7b show the expected locally parallel pattern. The structure at approximately half height of the box captures the boundary between the two essentially different particle behaviors described above. The structure in ellipse (B) in Fig. 7b separates particles moving from the back to the front and passing over the obstacle from those inverting their motion direction again. This separation is again the type our filter aims to focus on.

5 Computational Issues

Although the steps that are needed to compute the proposed filter are in theory rather straightforward, the application to discrete data offers some challenges we want to discuss. Namely, we address (a) the influence of the used sampling of the path line, and (b) the computational cost of computing the FTLE field following our suggestions compared to the standard algorithm.

5.1 The Impact of the Sampling

Our sampling of the path line at $\frac{1}{2}$, $\frac{3}{4}$, $\frac{7}{8}$ and the full integration time puts emphasis on the end of the path lines. Visually, this is intuitive, since we perceive path lines as parallel if their ends show this behavior. We anticipate common "spatial fate." Therefore will rather small direction changes toward the end of the registered path

Table 1 Error analysis of sampling density

		$N = $ 2	4	6	8	10
$\mathbf{v}_4$	Mean	1.2×10^{-3}	6.13×10^{-4}	3.23×10^{-6}	1.64×10^{-5}	3.5×10^{-5}
	Variance	1.27×10^{-6}	3.04×10^{-7}	8.42×10^{-7}	2.17×10^{-8}	3.5×10^{-9}
Double gyre	Mean	0.2	0.1	0.07	0.04	0.01
	Variance	0.03	0.01	0.03	9.4×10^{-4}	1.4×10^{-4}
Breaking dam	Mean	1.27×10^{-2}	5.47×10^{-3}	2.57×10^{-3}	9.82×10^{-4}	6.6×10^{-5}
	Variance	1.13×10^{-5}	2.4×10^{-7}	5.36×10^{-8}	7.7×10^{-9}	2.97×10^{-9}

lines intuitively be read as diverging behavior, since we anticipate that the motion will continue in the same direction.

We computed the separation measure for some of the data sets, namely the time-dependent ones, for $N = 2, 4, 6, 8, 10$ and compared the results point-wise, using the N=25 as reference value. Table 1 shows the average relative error and the variance in the computed fields. We chose to investigate the time-dependent data sets since this is most relevant in practice because FTLE computations for steady fields are usually avoided using VFT instead.

5.2 FTLE as Eigenvalues of the Cauchy-Green Tensor vs. Singular Values of the Deformation Gradient Tensor

Our filtering needs, in addition to the FTLE field, the left-singular vectors of the deformation gradient tensor. This is not a part of the usual algorithm to compute FTLE. However, the computation of the deformation gradient tensor is. Therefore, we do an informal comparison of the expected computational cost. Essentially, the here used alternative FTLE computation methods differ from the standard method in one aspect only: the use of the SVD instead of the eigenvalue decomposition. Standard algorithms for both decompositions are based on the same transformation in the iteration steps and have therefore the same complexity order. The singular matrices are an by-product of the SVD computation and do not need to be computed separately. For details we refer to Gill et al. [9]. Hence, computing the FTLE plus left-singular vectors will not be substantially slower than the usual computation of FTLE from the Cauchy-Green tensor. With the Maple implementations of SVD and eigenvalue decomposition the ratio of the computation time using the SVD to the time used with the standard formula is in the range [0.95, 1.06], i.e., the SVD-based method is in the worst case 6% slower than the standard method on the double gyre data set. In SimVis we used the linear algebra library *JAMA* (http://math.nist.gov/tnt/overview.html), which gives a ratio of 1.12 for the bursting dam data set, i.e., a 12% computational overhead. It is worthwhile noticing that our methods provides both the regular FTLE field plus the additional information needed to perform the filtering at once. Hence, ridge extraction algorithms may be applied as well, if wanted.

6   Discussion and Future Work

We examine the results from analyzing several different flow scenarios with the here proposed filtering scheme. We assess the filtered structures by seeding path lines in the unfiltered field and comparing the result of our filtering scheme to the result that we would expect from the path lines. In all cases the paths lines seeded in the filtered region show the expected locally parallel flow pattern (which we see as a satisfying confirmation of our more theoretical considerations with respect to the design of the proposed filter).

The computation of the flow map is, as expected, the bottle neck when applying our filtering to data sets. A speed-up of this computation could be achieved by exploiting the inherent parallel nature of path line computation and multi-core architectures. AMR and advection based methods to speed-up computations do not seem to be suitable at the first sight, since we are not extracting ridges and we do not know whether the structures that our filtering reveals have properties corresponding to material lines and surfaces. We intend to perform computational experiments to assess this question.

Finally, we intend to assess the effects of combining our filtering with other flow feature detectors. FTLE is known to miss out on some features as, for example, vortices. Hence, the combination of feature detectors is a promising approach [2, 3]. We have implemented our filtering in the SimVis framework [4], that is inherently suitable for the proposed investigation due to its combination of interactive visual analysis and 3D context visualization designed for flow data.

FTLE based methods, and consequently also our filtering of the field, are known to be heavily dependent on the choice of the integration length [8, 29]. Hence, the search for separation measures that can handle diverging and re-converging flow (as in flow around an obstacle) and similar behavior seems appealing.

7   Conclusion and Acknowledgments

In this paper we discuss two different types of separation and showed how to distinguish them filtering *FTLE*. Analyzing different flow scenarios, we showed that this distinction indeed yields a deeper understanding of separation structures. Separation is an important aspect in flow analysis and further classification of different types of this phenomenon seems to be a promising research direction.

References

1. Asimov, D.: Notes on the topology of vector fields and flows. Tech. rep., NASA Ames Research Center (1993). RNR-93-003

2. Bürger, R., Muigg, P., Doleisch, H., Hauser, H.: Interactive cross-detector analysis of vortical flow data. In: Proceedings of the 5th International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007). IEEE Computer Society, Zurich, Switzerland, pp. 98–110 (2007)

3. Bürger, R., Muigg, P., Ilčík, M., Doleisch, H., Hauser, H.: Integrating local feature detectors in the interactive visual analysis of flow simulation data. In: Museth, K., Möller, T., Ynnerman A. (eds.) Data Visualization 2007: Proceedings of the 9th Joint Eurographics – IEEE VGTC Symposium on Visualization (EuroVis 2007). Eurographics Association, Norrköping, Sweden, pp. 171–178. A K Peters (2007)

4. Doleisch, H.: SimVis: Interactive visual analysis of large and time-dependent 3D simulation data. In: Proceedings of the 2007 Winter Conference on Simulation (WSC 2007), pp. 712–720 (2007)

5. Doleisch, H., Hauser, H.: Smooth brushing for focus+context visualization of simulation data in 3D. J. WSCG **11**(1–2), 147–154 (2001)

6. Fuchs, R., Kemmler, J., Schindler, B., Waser, J., Sadlo, F., Hauser, H., Peikert, R.: Toward a lagrangian vector field topology. Comput. Graph. Forum **29**(3), 1163–1172 (2010)

7. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Vis. Comput. Graph. **13**(6), 1464–1471 (2007)

8. Garth, C., Li, G.S., Tricoche, X., Hansen, C.D., Hagen, H.: Visualization of coherent structures in transient 2D flows. In: Hege, H.C., Polthier, K., Scheuermann G. (eds.) Topology-Based Methods in Visualization II: Proceedings of the 2nd TopoInVis Workshop (TopoInVis 2007), pp. 1–13 (2009)

9. Gill, P.E., Murray, W., Wright, M.: Numerical Linear Algebra and Optimization, 1st edn. Addison Wesley Publishing Company, Boston MA, USA (1991)

10. Globus, A., Levit, C., Lasinski, T.: A tool for visualizing the topology of three-dimensional vector fields. In: Proceedings of IEEE Visualization '91. IEEE Computer Society, San Diego, California USA, pp. 33–40 (1991)

11. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins Studies in Mathematical Sciences. The Johns Hopkins University Press (1996)

12. Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I, 2nd edn. Springer Series in Computational Mathematics. Berlin Heidelberg, Germany, Springer (1993)

13. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Physica D **149**, 248–277 (2001)

14. Haller, G.: Lagrangian coherent structures from approximate velocity data. Phys. Fluids **14**, 1851–1861 (2002)

15. Haller, G., Sapsis, T.: Lagrangian coherent structures and the smallest finite-time lyapunov exponent. Chaos **21**(21), pp. 1–7, (2010)

16. Hayes, M.: On strain and straining. Arch. Rational Mech. Anal. **100**(3), 265–273 (1988)

17. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. IEEE Computer **22**(8), 27–36 (1989)

18. Helman, J., Hesselink, L.: Visualizing vector field topology in fluid flows. IEEE Comput. Graph. Appl. **11**, 36–46 (1991)

19. Jordan, D.W., Smith, P.: Nonlinear ordinary differential equations : an introduction for scientists and engineers, 4th edn. Oxford Applied and Engineering Mathematics. Oxford University Press, Oxford, UK (2007)

20. Kasten, J., Petz, C., Hotz, I., Noack, B., Hege, H.C.: Localized finite-time Lyapunov exponent for unsteady flow analysis. In proceedings of VMV'09 Braunschwag, Germany, pp. 265–274 (2009)

21. Lipinski, D., Mohseni, K.: A ridge tracking algorithm and error estimate for efficient computation of Lagrangian coherent structures. Chaos **20**(1), 017,504 (2010)
22. Mase, G.E.: Continuum Mechanics, 1st edn. Schaum's Outline Series. McGraw-Hill, New York NY, USA (1969)
23. Obermaier, H., Hering-Bertram, M., Kuhnert, J., Hagen, H.: Volume deformations in grid-less flow simulations. Comput. Graph. Forum **28**(3), 879–886 (2009)
24. Perry, A., Chong, M.: Topology of flow patterns in vortex motions and turbulence. Appl. Sci. Res. **53**, 357–374 (1994)
25. Pobitzer, A., Peikert, R., Fuchs, R., Schindler, B., Kuhn, A., Theisel, H., Matković, K., Hauser, H.: On the way towards topology-based visualization of unsteady flow – the state of the art. In: Eurographics 2010 – State of the Art Reports. Eurographics Association, Norrköping, Sweden (2010)
26. Sadlo, F., Peikert, R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. IEEE Trans. Vis. Comput. Graph. **13**(6), 1456–1463 (2007)
27. Sadlo, F., Peikert, R.: Visualizing Lagrangian coherent structures: a comparison to vector field topology. In: Hege, H.C., Polthier, K., Scheuermann G. (eds.) Topology-Based Methods in Visualization II: Proceedings of the 2nd TopoInVis Workshop (TopoInVis 2007), pp. 15–29, Grimma, Germany (2009)
28. Sadlo, F., Peikert, R.: Time-dependent visualization of Lagrangian coherent structures by grid advection. In: Pascucci, V., Tricoche, X., Hagen, H., Tierny J. (eds.) Topological Methods in Data Analysis and Visualization: Theory, Algorithms and Applications. Berlin Heidelberg, Germany, Springer (2011)
29. Shadden, S., Lekien, F., Marsden, J.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. Physica D **212**, 271–304 (2005). DOI 10.1016/j.physd.2005.10.007
30. Shi, K., Theisel, H., Weinkauf, T., Hege, H.C., Seidel, H.P.: Visualizing transport structures of time-dependent flow fields. Comput. Graph. Appl. **28**(5), 24–36 (2008)
31. Theisel, H., Weinkauf, T., Hege, H.C., Seidel, H.P.: Topological methods for 2D time-dependent vector fields based on stream lines and path lines. IEEE Trans. Vis. Comput. Graph. **11**(4), 383–394 (2005)
32. Wasberg, C.E.: Post-processing of marginally resolved spectral element data. In: Rønquist E.M. (ed.) ICOSAHOM Conference Proceedings. Berlin Heidelberg, Germany, Springer (2011)
33. Wiebel, A., Chan, R., Wolf, C., Robitzki, A., Stevens, A., Scheuermann, G.: Topological flow structures in a mathematical model for rotation-mediated cell aggregation. In: Pascucci, V., Tricoche, X., Hagen, H., Tierny J. (eds.) Topological Methods in Data Analysis and Visualization: Theory, Algorithms and Applications. Berlin Heidelberg, Germany, Springer (2011)

A Variance Based FTLE-Like Method for Unsteady Uncertain Vector Fields

Dominic Schneider, Jan Fuhrmann, Wieland Reich, and Gerik Scheuermann

1 Introduction

Uncertainty in vector field data poses a major challenge for visualization in general [16] but especially for the identification of coherent structures. For deterministic steady data *vector field topology* (VFT) reveals the overall structure in a condensed abstract view. However, VFT as such, is directly applicable only to steady or quasi stationary vector fields. This is due to the fact, that the theoretical foundation of VFT is build on stream lines. A concept build around path lines rather than streamline is the *finite-time Lyapunov exponent* (FTLE) which has its roots in dynamical systems theory and was introduced by Haller [13]. *Lagrangian coherent structures* (LCS), which can be extracted as ridge lines in the FTLE field, act as material lines or surfaces in the flow [19]. Hence they are either attracting if nearby particles converge towards them or repelling if they diverge from the respective LCS. They are the time-dependent analog to stable and unstable manifolds in steady vector fields. FTLE is one of the most important techniques to analyze flow structures in vector fields, however, it cannot be applied to uncertain vector fields directly. Haller [14] provided a formal analysis of FTLE behavior in the presence of error and found that LCS are very robust in the presence of these errors. In contrast, we propose an alternative, practical, variance-based approach directly applicable to stochastic flow maps computed from uncertain vector fields.

D. Schneider (✉) · W. Reich · G. Scheuermann
University of Leipzig, 04109 Leipzig, Germany
e-mail: schneider@informatik.uni-leipzig.de; reich@informatik.uni-leipzig.de; scheuermann@informatik.uni-leipzig.de

J. Fuhrmann
University of Heidelberg, Institute for Applied Mathematics, D-69120 Heidelberg, Germany
e-mail: jan.fuhrmann@uni-hd.de

2   Related Work

In this section we give an overview of relevant work on VFT, FTLE and work in the context of visualizing uncertainty in various types of data.

VFT has been introduced to visualization by Helman and Hesselink [15] and Globus et al. [10]. It aims at extracting so-called invariant sets. An invariant set is a special set of streamlines, most importantly isolated zeros (critical points) which are degenerate stream lines. Löffelmann et al. [22] visualized periodic orbits and Wischgoll et al. [38] and Chen et al. [4] presented an algorithm locating them. Invariant sets can segment the vector field into regions of similar flow, hence these invariant sets are termed *separatrices*. Displaying all separatrices, however, would lead to occlusion problems. A solution to this problem is the display of their intersection curves, the so-called saddle connectors [36].

For the topological analysis of time-dependent vector fields Sadlo et al. [32] used FTLE and generalized VFT, where degenerate streak lines take on the role of critical points. Weinkauf et al. [37] integrate a streak line field to facilitate time-dependent topology extraction. In contrast to the integration based perspective, Fuchs et al. [6] provide a differentiation based perspective to find critical points for time-dependent vector fields.

In visualization LCS have been increasingly subject of interest in the last decade [28]. Garth et al. visualized the FTLE field for 2D flows [9] using color and height maps and for 3D flows [7] using direct volume rendering. In subsequent work [8] FTLE has been used to identify attachment and separation on the surface of obstacles. Sadlo et al. compared VFT and LCS visualizations [31] and proposed a scheme for an accelerated computation [33].

Griethe at al. [11], Johnson et al. [17] and Pang et al. [27] give an overview of different uncertainty concepts and the different techniques used in visualization.

Uncertainty in vector fields has been visualized using additional geometry such as glyphs to represent the uncertainty at certain positions [20,21,23,39,40] to convey the amount of uncertainty explicitly. Other work used the concept of fuzziness or blurring to convey uncertainty in volumetric data [5, 30] or isosurfaces [12]. Lundstrom et al. [24] and Brown [3] used animation of the different possibilities to convey uncertainty information. Sanderson et al. [34] utilized reaction-diffusion systems to visualize vector fields and show that it is possible to incorporate uncertainty into their model. Botchen et al. [1, 2] proposed texture based flow visualization techniques and convey uncertainty by blurring streak lines using cross advection and diffusion.

Otto et al. [26] considered global uncertainty by the transport of local uncertainty in steady 2D flow fields. The domain is super-sampled by a high number of particles. Each particle is integrated and the final distribution is interpreted as a discrete particle density function. In contrast, we consider the original grid and analyze the transport of uncertainty by computing a stochastic flow map for unsteady flows. The stochastic flow map is analyzed using a Principal Component Analysis (PCA) yielding a scalar field showing FTLE-like structures for uncertain vector fields.

3 Uncertain Vector Fields

In this work we consider steady and unsteady 2D vector fields. In the following these are called *deterministic vector fields*. In contrast, *uncertain vector fields* no longer map a position to a single unique vector but rather to a probability distribution of vectors. We adopt the definition for a steady uncertain 2D vector field given in [26].

We follow the approach of [14] and examine a stochastic differential equation describing the vector field and a chosen error model. Since we have full control of the amount and type of error, this provides us with a method to analyze the vector field in the presence of uncertainty, i.e. error. The stochastic differential equation is solved by stochastic integration which is described in the next section.

3.1 Stochastic Integration

In the following we develop a model for stochastic integration in a vector field $\mathbf{v}$ defined over a domain D. We start with solving the following classical ordinary differential equation (ODE):

$$d\phi = \mathbf{v}(\phi(t), t)dt \qquad \phi(t_0) = x_0 \tag{1}$$

where ϕ is a map $D \to D$. Since we want to analyze vector fields in the presence of errors, it seems reasonable to modify (1) to include random effects disturbing the system, thus turning it into a *stochastic differential equation* (SDE):

$$d\Phi = \mathbf{v}(\Phi(t), t)dt + B(\Phi(t))d\xi_t \qquad \Phi(t_0) = x_0 \tag{2}$$

The first term of the right-hand side resembles the classical formulation (see (1)) and the second term represents the disturbance with $d\xi_t$ being a so-called *continuous-time stochastic process*, where ξ_t is indexed by real numbers $t \geq 0$ and $B(.)$ characterizing the disturbance.

The most popular example of a stochastic process that is ubiquitous in physics, chemistry, finance and mathematics is the *Wiener process* W_t named after Norbert Wiener with the following three properties (see [35]):

Property 3. For each t, the random variable W_t is normally distributed with mean 0 and variance t.

Property 4. For each $t_1 \leq t_2$, the normal random variable $W_{t_2} - W_{t_1}$ is independent of the random variable W_{t_1}, and in fact independent of all W_t, $0 \leq t \leq t_1$.

Property 5. The Wiener process W_t can be represented by continuous paths.

In order to simplify the matter we set $B(\Phi(t)) = \varepsilon$ constant. This leads to the following stochastic differential equation:

$$d\Phi = \mathbf{v}(\Phi(t), t)dt + \varepsilon dW_t \qquad \Phi(t_0) = x_0 \qquad (3)$$

Now we can solve (3) in the following way:

$$\Phi(t) = x_0 + \int_0^t \mathbf{v}(\Phi(\tau), \tau)d\tau + \int_0^t \varepsilon dW_\tau \qquad (4)$$

This resembles a so-called *drift and diffusion* model where $\mathbf{v}$ is referred to as the drift coefficient, while ε is called the diffusion coefficient. A helpful interpretation of the stochastic integral in (4) is that in a time interval of length 1 the stochastic process changes its value by an amount that is normally distributed with expectation $\mathbf{v}$ and variance ε. This change is independent of the processes past behavior because the increments of the Wiener process are independent and normally distributed (see property 1–3). The integral of the Wiener process, in particular, yields a diffusion term and as necessary a contribution to the drift term. For an in depth discussion of stochastic differential equations and integrals we refer the reader to [29].

3.2 Error Model

Despite the popularity and importance of the Wiener process in other fields of research we will use a different random process. We deem this necessary due to the fact that the Wiener process has normally (Gaussian) distributed increments. This implies that the increments are unbounded meaning an arbitrary large error could occur. This means the numerical stochastic integration could perform steps of arbitrary length. This does not account for the situation arising with CFD data, because it would mean that the simulation contains arbitrary large errors. Instead, an error bound is provided describing the maximal error of the data. Moreover, we do not want to make any assumption about the error distribution within this bound. This leads us to an equal distribution with zero mean, bounded by a *n*-dimensional ball with radius ε. Furthermore, we assume independence for this stochastic process (see property 2 of the Wiener process). On the contrary, the error in a CFD simulation at one grid point is likely not to be independent from neighboring grid points. However, the modeling of this dependence would require exact knowledge of the underlying CFD solver and it would be a highly complex task. Hence, the presented error model is a rather pragmatic approach.

Furthermore, we assume the same error bound ε for the whole domain. This has the effect of a homogeneous error distribution and the error being independent of the location in space and time. Formally this is achieved by setting $B(\Phi(t)) = \varepsilon$ constant (see Sect. 3.1) which leads to the following SDE:

$$d\Phi = \mathbf{v}(\Phi(t), t)dt + \varepsilon dR_t \qquad \Phi(t_0) = x_0 \qquad (5)$$

with dR_t denoting the chosen random process.

If the error distribution, however, turns out to be inhomogeneous over time, ε is no longer a constant but will depend on the location in space and time $\varepsilon(\Phi(t), t)$. Moreover, the error ε does not need to be a scalar quantity but can provide directional information, hence becoming biased or anisotropic. In these cases the underlying SDE needs to be modified to fit the chosen error model resulting in a different numerical approximation (see Sect. 3.3). In general the error model can always be adjusted to fit ones needs.

3.3 Numerical Approximation

The classical numerical approximation schemes (e.g. Euler or Runge–Kutta) cannot be applied to a SDE as such, but needs to be modified. In order to approximate the stochastic integration process numerically we need to discretize the stochastic process dR_t in (5). This is accomplished by the Euler–Maruyama method (see [18]) which provides an approximate numerical solution of a SDE. Application to the stochastic process dR_t yields

$$\Delta R_n = \varepsilon \, \Delta t \, RW \qquad (6)$$

where RW denotes an increment of a so-called *random walk*. A random walk is the mathematical formalization of a trajectory consisting of successive random steps discretizing the considered stochastic process. Our error model (see Sect. 3.2) requires RW to be an undirected random walk with a bounded symmetric uniform probability distribution with bound ε. The discretization of (5) then reads as follows:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \, \mathbf{v}(\mathbf{y}_n, t) + \Delta t \, \varepsilon \, RW \qquad (7)$$

As one can see, the classical Euler method is a special case of the Euler–Maruyama integration for $\varepsilon = 0$. The successive application of (7) yields a stochastic stream- or path-line, respectively. However, the obtained trajectory is an approximate realization of the solution stochastic process Φ in (5) and each one will be different.

In the case that the vector field consists of measured data and a stochastic modeling of the process is not possible then the uncertain vector field $\mathbf{v}_u$, consisting of the measured data, can of course be evaluated directly yielding

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \, \mathbf{v}_u(\mathbf{y}_n, t) \qquad (8)$$

where $\mathbf{v}_u$ is automatically respecting the probability distribution.

3.4 Stochastic Flow Maps

A classical discrete flow map $\phi_{t0}^{t_0+t}(\mathbf{x})$ maps from a sample position $\mathbf{x} \in D$ to the position of a particle started at $\mathbf{x}$ at time t_0 advected by the flow for the time t. In other words $\phi_{t0}^{t_0+t}(\mathbf{x})$ maps $\mathbf{x}$ to its advected position. In case the integration reaches the boundary we store the position on the boundary in the flow map. Classical flow maps are computed by integrating one particle per sample position. In contrast, *stochastic flow maps* Φ store a whole distribution per sample position, yet the principle is the same. We approximate the distributions in the stochastic flow map by sampling: we start multiple stochastic integrations at each grid point $\mathbf{x}$. The number of integrations per position is prescribed by the parameter N.

In Sect. 4 we evaluate the stochastic flow map at a certain grid point r and at the neighboring grid points. Hence the particles started from these positions are part of the same random experiment. Assigning two or more particles to the same random experiment means they are dependent. This dependency manifests in the way that if two particles meet at exactly the same point in space and time they experience the same fluctuation, i.e. RW in (5) is for both particles the same. However, if one of these two particles is only a small distance off that position the random variables are independent, i.e. RW evaluates for each particle to a different value.

An algorithm taking this into account would create N deterministic vector fields $\mathbf{v}_d$, each disturbed according to the error model. For each of these vector fields particle integration is carried out deterministically. This way we obtain a distribution consisting of advected particles for each start point as well, at the expense of additional memory consumption for the creation of additional vector fields and the computational cost for their construction.

Now we argue that the probability of any two particles which are part of the same random experiment seeded at different positions in space meet at exactly the same position in space and time is zero in the continuous setting. Since we are in a discrete setting the probability of this event is not zero anymore. However, if this still very unlikely event is happening we argue that in a continuous setting both particles would not have met at exactly the same position in space and time. Hence we argue further that this event is due to the discretization of the vector field and especially due to the discrete approximation of the integration process. The conclusion of this argumentation is that we calculate trajectories for particles of the same random experiment simply as they were independent random experiments since the possible gain is negligible. After all we are only interested in the expected value and the variance of the particle distribution. Note that this argumentation is only valid for an error model assuming independence of errors.

4 FTLE-Like Variance Based Analysis of Stochastic Flow Maps

The classical FTLE method [13] measures the maximal separation or expansion rate of two closely seeded particles when advected by the flow for a finite time t. FTLE can be computed by utilizing the above mentioned flow map. More precisely it is

the square root of the spectral norm of the (right) Cauchy-Green deformation tensor $\Delta(\mathbf{x})$ [25]:

$$\Delta(\mathbf{x}, t, t_0) = (\nabla\phi_{t0}^{t_0+t}(\mathbf{x}))^T (\nabla\phi_{t0}^{t_0+t}(\mathbf{x})) \tag{9}$$

The function Δ is a symmetric matrix and measures the square of the distance change due to deformation. Now FTLE is defined as the logarithm of the Cauchy–Green deformation tensor's maximum eigenvalue λ_{max} normalized by the absolute advection time t. More formally this reads as follows:

$$FTLE_{t_0}^t(\mathbf{x}) = \frac{1}{|t|} \ln \sqrt{\lambda_{max}(\Delta(\mathbf{x}, t, t_0))} \tag{10}$$

For an in depth discussion the reader is referred to the work of Haller [13].

The above description provides us with an algorithm for the computation of FTLE in deterministic vector fields. However, it cannot be directly applied to uncertain vector fields since we are dealing with probability distributions. On the other hand, the idea of FTLE is to find the maximal stretching a virtual particle experiences during its lifetime. We think this idea can be carried over to the realm of uncertain vector fields by replacing the stretching with variance. Therefore, we propose a new geometry driven technique to compute a FTLE-like field for uncertain vector fields based on variance. The FTLE principle then translates to finding the maximal variance of the advected distribution. The best linear approximation for this problem is provided by the PCA. PCA is hence used to measure the deformation of the seeding points and the according probability distributions comprising of the advected particles. We evaluate the stochastic flow map at the current and neighboring positions wrt. the mesh. This is necessary mainly for two reasons: First, we need a reference value for the variance to measure the stretching. Moreover this simulates the discrete derivation process in the FTLE computation, since the direct neighbors are involved in the numerical approximation of the derivative. Second, if we would consider only the current position and the error ε is rather small the visualization would heavily suffer from aliasing effects.

The distributions gained by evaluating the stochastic flow map are approximated by stochastic trajectory endpoints. All these endpoints are merged to one single set or distribution respectively. From this set we compute the covariance matrix C which is a symmetric matrix like the Cauchy–Green deformation tensor. The matrix C is a linear model measuring the square of the standard deviation (variance) in every direction of space. We are interested in the maximal variance of C which is represented by the maximal eigenvalue. Figure 1 summarizes the algorithm visually. The relative maximal standard deviation consists of the maximal standard deviation of the matrix C divided by the maximal standard deviation of the seed points. If we interpret the relative maximal standard deviation as a measure for the maximal stretching of a distribution after integration in the uncertain vector field and recall that variance is the squared standard deviation and account for the similarities between the Cauchy–Green deformation tensor and the covariance matrix C we can simply rewrite (10) by changing Δ and C:
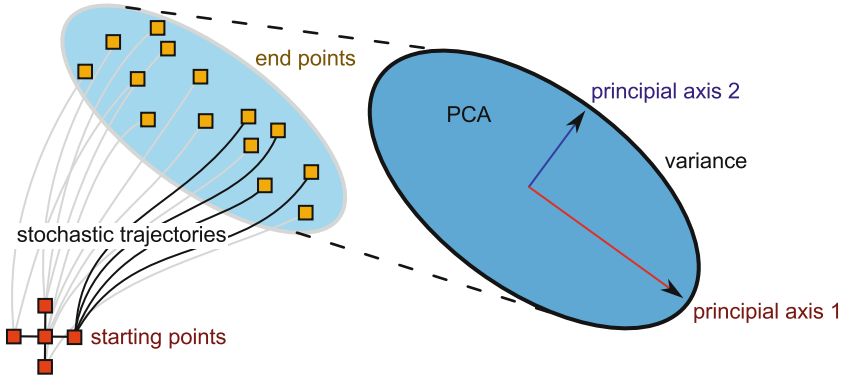
Fig. 1 Stochastic integration from a starting point gives a distribution of end points due to uncertainty. A principal component analysis of the start and end point distribution provides information about the maximum amount of stretching

$$FTVA_{t_0}^t(\mathbf{x}) = \frac{1}{|t|} \ln \sqrt{\frac{\lambda_{max}(C(\mathbf{x}, t, t_0))}{\lambda_{max}(C(\mathbf{x}, t_0, t_0))}} \qquad (11)$$

where FTVA stands for *finite time variance analysis*.

5   Results

In this section we apply our method to a 2D steady and unsteady vector field to demonstrate the utility and robustness of our method. The quality of the results naturally depends on certain parameters, which are integration time, the amount of error and the number of particles per position.

5.1   Tilted Bar

Our first dataset is a 2D unsteady vector field comprising of 100 time steps each with 79,200 positions and 78,421 quad cells. It consists of a Karman vortex street behind a tilted bar. A direct visual comparison between FTLE and FTVA in the steady case of a selected time step with integration time 1 is depicted in Fig. 2 and shows only slight visible differences. We used the color mapping proposed in [9], where red structures indicate high divergence in positive time, blue high divergence in negative time, black high divergence in both time directions and white no divergence.

In order to show the robustness of our method and the computed LCS, we increased the error in the computations successively. Figure 3 shows LCS for integration time 0.4 but with different amounts of error. An expected result is that the
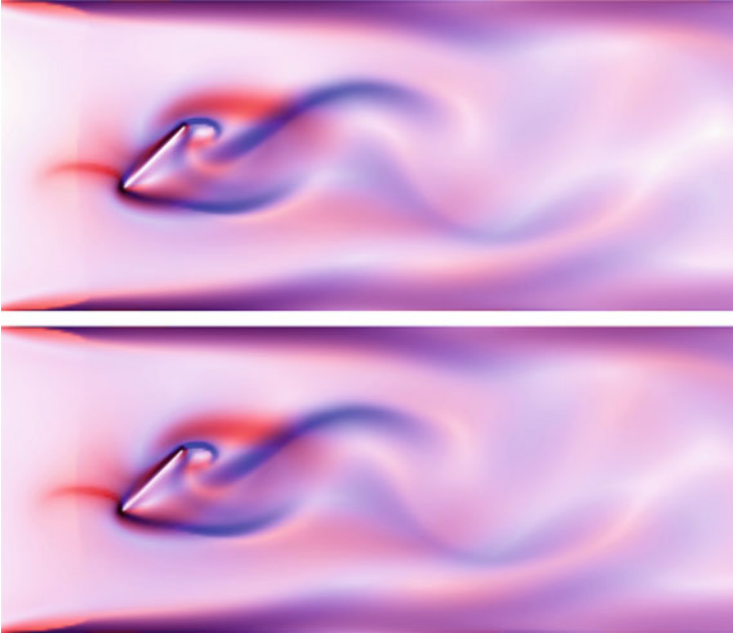
Fig. 2 Visual comparison between FTLE and FTVA computed for the tilted bar dataset with integration time 1.0. *Top*: FTLE field. *Bottom*: FTVA field with one particle per position and $\varepsilon = 0$

strength of LCS weakens as the error grows (see Fig. 3, bottom). However, despite the weakening, LCS are very robust. They remain visible for a surprisingly large error which, according to our model, grows linearly with the integration time.

Another parameter to be studied is the amount of particles per position which is a crucial one in terms of quality. For the visualizations in Fig. 4 the number of particles per position has been set to 100 and the error has been varied. As a result, if too few particles are chosen the distribution cannot be approximated correctly. Hence the FTVA fields become disturbed, which is manifested in an unsmooth color map. As expected, this effect unfolds with increasing error.

5.2 Turbulent Jet Flow

Our second dataset is a swirling jet flow entering the domain, containing resting fluid, to the left. The dataset is steady, consist of 124×101 quad cells and is highly turbulent. Again, there is almost no visible difference between a visualization of the FTLE and the FTVA fields (see Fig. 5). As was the case in the former example dataset the strength of the LCS in the FTVA fields weakens and become less sharp
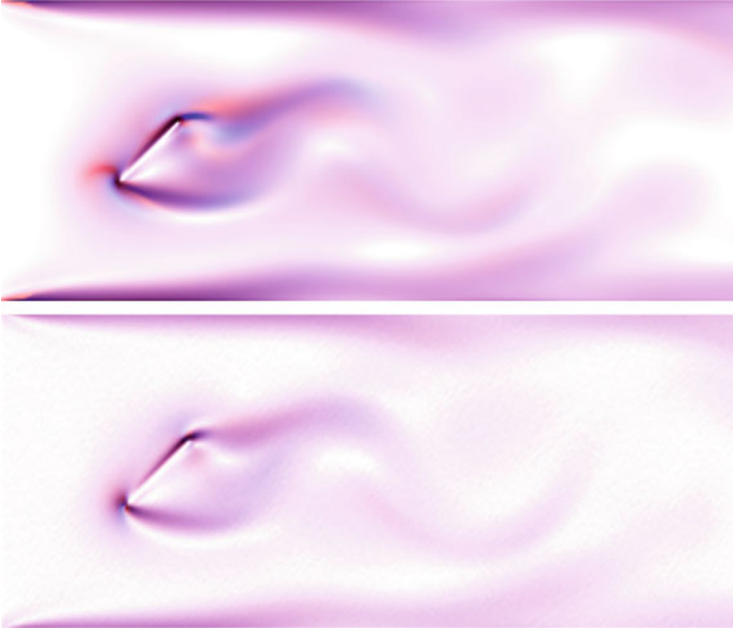
Fig. 3 Weakening of LCS in the titlted bar dataset for growing error, for an integration time of 0.4. *Top*: 10 Particles and $\varepsilon = 0.01$. *Bottom*: 500 particles and $\varepsilon = 1.0$

with growing error. However, even in the presence of large errors LCS remain visible (see Fig. 6).

The lion's share of the computation time is spent integrating particles for the stochastic flow map, which is about 99% of the timings given in Table 1. These timings are given for a non-parallelized version of the algorithm executed on an Intel Xeon CPU E5620 with 2.4 GHz.

6 Conclusion and Future Work

In this paper we have proposed a method to compute an FTLE-like measure called FTVA (finite time variance analysis) to find regions of converging and diverging flow in uncertain flow fields. We have produced promising visualization results, however, we would like to study application examples with naturally arising uncertain vector fields. Despite a successful implementation of the stochastic flow map further research is needed to limit the high computational cost (see Table 1). Therefore, we will look into an adaptive approach for the automatic determination of the number of stochastic trajectories necessary per position, which would be highly desirable. Furthermore, we want to research deeper into the differences and
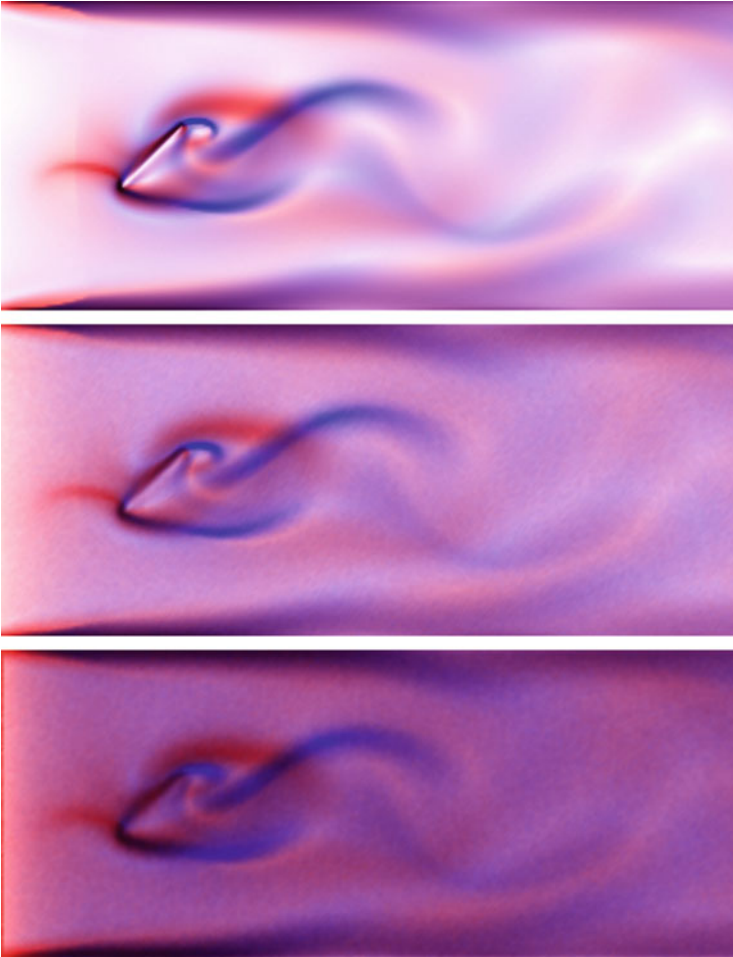
Fig. 4 Comparison between FTVA results for the tilted bar dataset for a constant number of particles (100 particles per position) to show the influence of error on the visualizations. *Top*: integration length 1.0 and $\varepsilon = 0.1$. *Middle*: integration length 1.0 and $\varepsilon = 1.0$. *Bottom*: integration length 1.0 and $\varepsilon = 2.0$

similarities of the covariance matrix and the Cauchy–Green stress tensor and exploit the results to improve our method. Also a parallelized version should be much faster and additionally acceleration techniques [7, 33] can be implemented as well.

The extension of the presented concepts to 3D is straight forward: first: extend stochastic flow map computation to 3D, second: compute stochastic flow map for every grid point, third: compute PCAs in 3D. Neither of which poses a major hurdle.
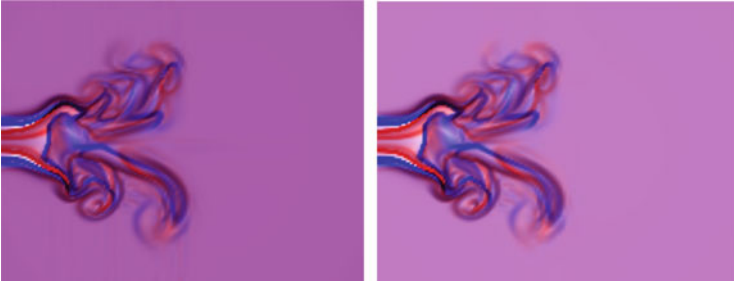
Fig. 5 Comparison between FTLE (*left*) and FTVA (*right*) with one particle per position and $\varepsilon = 0$ for the jet flow dataset
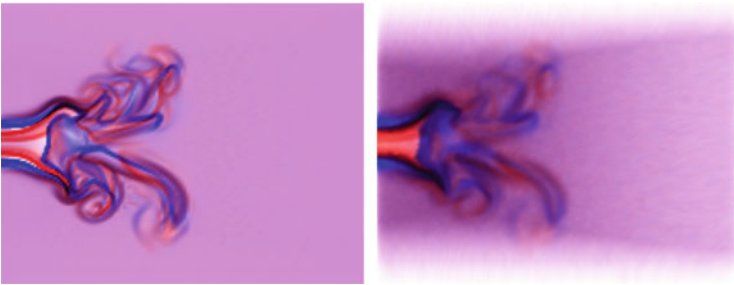
Fig. 6 Visualizations for the jet flow dataset showing the blurring of the FTVA structures for integration length 1,100 particles per position and varying error. *Left*: $\varepsilon = 1$. *Right*: $\varepsilon = 5$

Table 1 Computation times for the tilted bar dataset for different number of particles per position

No. particles per position	No. particles total	Computation time
10	7.92×10^5	7 min
100	7.92×10^6	1 h 15 min
500	3.96×10^7	6 h 18 min
1,000	7.92×10^7	12 h 41 min

Timings are given for non-parallel integration.

References

1. Botchen, R.P., Weiskopf, D.: Texture-based visualization of uncertainty in flow fields. In: Proceedings of IEEE Visualization '05, Minneapolis, MN, pp. 647–654 (2005)
2. Botchen, R.P., Weiskopf, D., Ertl, T.: Interactive visualization of uncertainty in flow fields using texture-based techniques. In: Electronic Proceedings of 12th International Symposium on Flow Visualization '06, Göttingen, Germany (2006)

3. Brown, R.: Animated visual vibrations as an uncertainty visualisation technique. In: Proceedings of GRAPHITE, GRAPHITE '04, pp. 84–89. ACM, New York, NY, USA (2004)
4. Chen, G., Mischaikow, K., Laramee, R.S., Zhang, E.: Efficient morse decompositions of vector fields. IEEE Trans. Visual. Comput. Graph. **14**, 848–862 (2008)
5. Djurcilov, S., Kim, K., Lermusiaux,P., Pang, A.: Visualizing scalar volumetric data with uncertainty. Comput. Graph. **26**(2), 239–248 (2002)
6. Fuchs, R., Kemmler, J., Schindler, B., Sadlo, F., Hauser, H., Peikert, R.: Toward a Lagrangian vector field topology. Comput. Graph. Forum **29**(3), 1163–1172 (2010)
7. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Visual. Comput. Graph. **13**, 1464–1471 (2007)
8. Garth, C., Wiebel,A., Tricoche, X., Joy, K., Scheuermann,G.: Lagrangian visualization of flow-embedded surface structures. Comput. Graph. Forum **27**(3), 1007–1014 (2008)
9. Garth, C., Li, G.-S., Tricoche, X., Hansen, C.D., Hagen, H.: Visualization of coherent structures in transient 2D flows. In: Topology-Based Methods in Visualization II, pp. 1 – 13. Springer, Berlin (2009)
10. Globus, A., Levit, C., Lasinski, T.: A tool for visualizing the topology of three-dimensional vector fields. In: Proceedings of IEEE Visualization 91, San Diego, CA, pp. 33–40 (1991)
11. Griethe, H., Schumann, H.: The visualization of uncertain data: methods and problems. In: Proceedings of SimVis '06, Magdeburg, Germany, pp. 143–156 (2006)
12. Grigoryan, G., Rheingans, P.: Point-based probabilistic surfaces to show surface uncertainty. IEEE Trans. Visual. Comput. Graph. **10**, 564–573 (2004)
13. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional flows. Physica D **149**, 248 – 277 (2001)
14. Haller, G.: Lagrangian coherent structures from approximate velocity data. Phys. Fluids **A14**, 1851–1861 (2002)
15. Helman, J.L., Hesselink, L.: Visualizing vector field topology in fluid flows. IEEE Comput. Graph. Appl. **11**(3), 36–46 (1991)
16. Johnson, C.: Top scientific visualization research problems. IEEE Comput. Graph. Appl. **24**, 13–17 (2004)
17. Johnson, C.R., Sanderson, A.R.: A next step: visualizing errors and uncertainty. IEEE Comput. Graph. Appl. **23**, 6–10 (2003)
18. Kloeden, P.E., Platen, E.: Numerical Solution of Stochastic Differential Equations. Springer, Berlin, 1992.
19. Lekien, F., Coulliette, C., Mariano, A.J., Ryan, E.H., Shay, L.K., Haller, G., Marsden, J.E.: Pollution release tied to invariant manifolds: a case study for the coast of florida. Physica D, **210**(1–2), 1–20 (2005)
20. Li, H., Fu, C.-W., Li, Y., Hanson, A.: Visualizing large-scale uncertainty in astrophysical data. IEEE Trans. Visual. Comput. Graph. **13**, 1640–1647 (2007)
21. Lodha, S.K., Pang, A., Sheehan, R.E., Wittenbrink, C.M.: Uflow: visualizing uncertainty in fluid flow. In: Proceedings of the 7th Conference on Visualization '96, VIS '96, Los Alamitos, CA, USA, 1996, pp. 249–254. IEEE Computer Society Press.
22. Löffelmann, H., Kucera, T., Gröller, E.: Visualizing poincar maps together with the underlying flow. In: Hege, H.-Ch., Polthier, K. (eds.) Proceedings of the International Workshop on Visualization and Mathematics '97, Mathematical Visualization, Berlin-Dahlem, Germany, pp. 315–347. Springer, Heidelberg (1997)
23. Lopes, A., Brodlie, K.: Accuracy in 3D particle tracing. In: Mathematical Visualization, pp. 329–341. Springer, Berlin (1998)
24. Lundstrom, C., Ljung, P., Persson, A., Ynnerman, A.: Uncertainty visualization in medical volume rendering using probabilistic animation. IEEE Trans. Visual. Comput. Graph. **13**, 1648–1655 (2007)
25. Mase, G.T., Mase, G.E.: Continuum Mechanics for Engineers. CRC Press, Boca Raton, FL (1999)

26. Otto, M., Germer, T., Hege, H.-C., Theisel, H.: Uncertain 2D vector field topology. Comput. Graph. Forum **29**, 347–356 (2010).
27. Pang, A.T., Wittenbrink, C.M., Lodha, S.K.: Approaches to uncertainty visualization. Vis. Comput. **13**, 370–390 (1997)
28. Pobitzer, A., Peikert, R., Fuchs, R., Schindler, B., Kuhn, A., Theisel, H., Matkovic, K., Hauser, H.: On the way towards topology-based visualization of unsteady flow – the state of the art. In: EuroGraphics 2010 State of the Art Reports (STARs), Norrköping, Sweden, pp. 137–154 (2010)
29. Protter, P.E.: Stochastic Integration and Differential Equations. Springer, Berlin (2003)
30. Rhodes, P.J., Laramee, R.S., Daniel Bergeron, R., Sparr,T.M.: Uncertainty visualization methods in isosurface volume rendering. In: Eurographics 2003, Short Papers, Granada, Spain, pp. 83–88 (2003)
31. Sadlo, F., Peikert, R.: Visualizing lagrangian coherent structures and comparison to vector field topology. In: Topology-Based Methods in Visualization II, pp. 1 – 13. Springer, Berlin (2009)
32. Sadlo, F., Weiskopf, D.: Time-dependent 2-D vector field topology: an approach inspired by Lagrangian coherent structures. Comput. Graph. Forum **29**(1), 88–100 (2010)
33. Sadlo, F., Rigazzi, A., Peikert, R.: Time-dependent visualization of lagrangian coherent structures by grid advection. In: Topological Methods in Data Analysis and Visualization. Springer, Berlin (2009)
34. Sanderson, A.R., Johnson, C.R., Kirby, R.M.: Display of vector fields using a reaction-diffusion model. In: Proceedings of the Conference on Visualization '04, VIS '04, Washington, DC, USA, pp. 115–122. IEEE Computer Society (2004)
35. Sauer, T.: Numerical solution of stochastic differential equations in finance. In: Duan, J.-C., Härdle, W.K., Gentle, J.E.S. (eds.) Handbook of Computational Finance. Springer Handbooks of Computational Statistics, pp. 529–550. Springer, Berlin (2012)
36. Theisel, H., Weinkauf, T., Hege, H.-C., Seidel, H.-P.: Saddle connectors – an approach to 378 visualizing the topological skeleton of complex 3D vector fields. In: Proceedings of IEEE Visualization '03, Seattle, WA, pp. 225–232 (2003)
37. Weinkauf, T., Theisel, H.: Streak lines as tangent curves of a derived vector field. IEEE Trans. Visual. Comput. Graph. (Proceedings of Visualization 2010), 16(6):1225–1234 (2010)
38. Wischgoll, T., Scheuermann, G.: Locating closed streamlines in 3D vector fields. In: Proceedings of the Symposium on Data Visualisation 2002, VISSYM '02, Aire-la-Ville, Switzerland, Switzerland, pp. 227–232. Eurographics Association (2002)
39. Wittenbrink, C.M., Pang, A.T., Lodha, S.K.: Glyphs for visualizing uncertainty in vector fields. IEEE Trans. Visual. Comput. Graph. **2**, 266–279 (1996)
40. Zehner, B., Watanabe, N., Kolditz, O.: Visualization of gridded scalar data with uncertainty in geosciences. Comput. Geosci. **36**, 1268–1275 (2010)

On the Finite-Time Scope for Computing Lagrangian Coherent Structures from Lyapunov Exponents

Filip Sadlo, Markus Üffinger, Thomas Ertl, and Daniel Weiskopf

1 Introduction

Traditional vector field topology [7] deals with several types of distinguished streamlines. Those that degenerate to isolated points play a special role in the form of critical points: if they exhibit saddle-type flow behavior in their linearized neighborhood, they give rise to separatrices. Separatrices are another type of distinguished streamlines: those that converge to saddle-type critical points in forward or reverse time. Other types include periodic orbits, i.e., isolated closed streamlines and invariant tori. Common to all these special cases is their derivation: these constructs are obtained as limit cases as integration time of the streamline approaches infinity [1]. This, except for technical issues, does not represent a problem. Steady vector fields are constant over time and hence do not require the notion of scope of time.

The traditional Lyapunov exponent (LE) shares many aspects with vector field topology. Despite additional technical issues regarding numerics and boundedness of the temporal domain, its properties are well defined and do not depend on additional parameters. It was, however, the boundedness of the temporal domain together with the aim of choosing a region of interest also in terms of time that led to the finite-time Lyapunov exponent (FTLE). A main risk, however, is the choice of too short advection time for FTLE computation and resulting misinterpretation.

Lagrangian coherent structures (LCS) by means of the FTLE [6] have become a prominent alternative for the investigation of time-dependent topology of vector fields. In this paper, we present a method for validating and choosing the advection time parameter for 2D FTLE computation with respect to prescribed error measures.

F. Sadlo (✉) · M. Üffinger · T. Ertl · D. Weiskopf
Visualization Research Center University of Stuttgart (VISUS), Germany
e-mail: sadlo@visus.uni-stuttgart.de; ueffinms@visus.uni-stuttgart.de;
ertl@visus.uni-stuttgart.de; weiskopf@visus.uni-stuttgart.de

We base our approach on a main principle of LCS: their advection property [12], i.e., their behavior as material lines.

1.1 Finite-Time Lyapunov Exponent

Vector fields exhibit a spectrum of Lyapunov exponents. It is the largest exponent in this spectrum that has become a prominent tool for predictability analysis in time-dependent vector fields. The LE can be determined by computing two neighboring trajectories in phase space and measuring their separation rate as time approaches infinity. Since the LE was originally introduced for predictability analysis, it has to reflect properties along trajectories. Therefore, precaution has to be taken to assure that the "neighboring" trajectories do not separate too far, e.g., by renormalization [2].

Since the systems under investigation are often defined on a finite temporal domain only, or because it is often the objective of the user to restrict the analysis to a temporal region of interest, the FTLE has been becoming more and more popular. Again, there are techniques to assure proximity of the (implicitly) involved trajectories, such as the localized FTLE [8].

Whereas the LE and FTLE have been applied for a long time to predictability analysis, there is a recent trend in the visualization community to use FTLE for revealing the topology of time-dependent vector fields. Haller [6] showed that ridges present in the FTLE represent a time-dependent counterpart to separatrices from vector field topology [7]: they separate regions of qualitatively different behavior.

In the context of time-dependent vector field topology, the FTLE is typically computed from the *flow map* $\phi_{t_0}^{t_0+T}(\mathbf{x})$, mapping seed points $\mathbf{x}$ of trajectories starting at time t_0 to their end points after advection for finite time T. According to Haller [7], the FTLE $\sigma(\mathbf{x}, t_0, T)$ is computed from the flow map ϕ as follows:

$$\sigma(\mathbf{x}, t_0, T) = \frac{1}{|T|} \ln \sqrt{\lambda_{\max}\left[\left(\nabla\phi_{t_0}^{t_0+T}(\mathbf{x})\right)^{\top} \nabla\phi_{t_0}^{t_0+T}(\mathbf{x})\right]}, \tag{1}$$

$\lambda_{\max}(\cdot)$ being the major eigenvalue.

2 Method

In this paper, we do not address the choice of the advection time parameter T in the context of specific applications and related application-oriented questions. In fact, we rather consider the fundamental advection property of LCS to find appropriate advection times inside a prescribed temporal interval of interest.
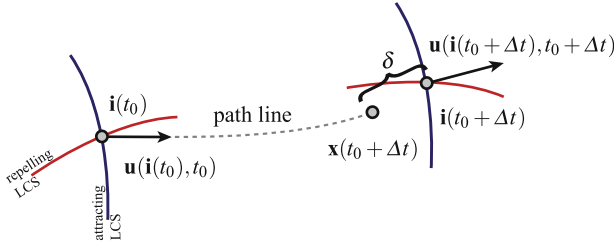
Fig. 1 FTLE ridge intersections **i** obtained with a fixed FTLE advection time T with starting times t_0 (*left*) and $t_0 + \Delta t$ (*right*). The spatial discrepancy δ is defined as the distance between $\mathbf{i}(t_0 + \Delta t)$ and the endpoint $\mathbf{x}(t_0 + \Delta t)$ of the path line started at $\mathbf{i}(t_0)$ after advection time Δt

Let us examine the two extreme choices for T: infinity and zero. As $T \rightarrow \infty$, the FTLE converges[1] to the classical LE. Hence, this extreme case imposes no particular concerns. The interpretation and use of the LE is well established. If, on the other hand, $T \rightarrow 0$, it can be easily shown that the FTLE converges to the largest eigenvalue of the rate of strain tensor of the vector field. In this case, due to the instantaneous property of the rate of strain tensor, it cannot be expected that the advection property [6] of ridges inside this field is met, consistent with Shadden et al.'s Theorem 4.4 [12].

Shadden et al. measured fluxes of the instantaneous velocity field across FTLE ridges as a means of verifying the advection property of given FTLE ridges. However, zero flux alone is a necessary but not sufficient condition for advection: it does not capture tangential motion, i.e., the motion component along the ridge cannot be assumed to satisfy the advection property even if the flux across the ridge is zero. Unfortunately, it is hard to identify point correspondences between ridges from FTLE fields with different starting times t_0 (Fig. 1) because common ridge definitions, such as height ridges by Eberly [3], are purely geometric, i.e., they are not represented by identifiable particles that advect. We therefore follow a different approach: we measure the advection property only for distinguished points on the FTLE ridges, i.e., we identify point correspondences in terms of advection.

For sufficiently well defined LCS (according to Shadden et al.), the advection property holds for both ridges in the FTLE field computed from forward trajectories and ridges in the FTLE field computed from reverse-time $(-T)$ trajectories. These ridges typically intersect. The intersection points give rise to hyperbolic trajectories [5, 11] and are important in the Lagrangian skeleton of turbulence [9]. Hence, if both types of ridges satisfy the advection property, this property must also hold for their intersections. In 2D vector fields, these intersections are isolated points (we exclude congruent ridge regions because these represent degenerate cases that

---

[1]We follow a conceptual, or continuous, argumentation here. In the discrete case, precautions have to be taken such as renormalization [2] or evaluation of the velocity gradient along the trajectories [8].

can be avoided by perturbing the vector field). Thus, one needs to identify point correspondence between ridge intersections of successive FTLE time steps (at t_0 and $t_0 + \Delta t$) (Fig. 1).

As FTLE computation typically depends on time-dependent discretized vector fields, it is very hard to derive generic rules for parametrizing FTLE visualizations. FTLE computation is heavily dependent on the structure and position of the sampling grid, and the advection time T. Furthermore, FTLE ridges only tend to represent LCS if they are sufficiently sharp [12]. Providing automatic strategies for finding a good choice of the parameters is very hard because many decisions directly depend on the goal of the user. Therefore, FTLE visualization, as other feature extraction procedures, is typically a trial-and-error procedure representing the basic exploration by visualization. Once the user has found a sampling grid that sufficiently captures the FTLE structures he or she wants to see, has found an appropriate threshold filtering out insufficiently sharp ridges (this can be accomplished by filtering by an eigenvalue of the FTLE Hessian [10]), and has found a minimum and maximum FTLE advection time T, our new technique takes over these parameters. Although our method can be further parametrized manually, it performs well with the default values (described below). As the result, our technique provides a plot of advection discrepancy with respect to FTLE advection time T. It can identify local and global optima inside the prescribed range of T, and in particular, provide a lower bound on T with respect to a prescribed accuracy in terms of average advection error of ridge intersections.

One contribution of this paper is a comparably robust technique for tracking FTLE ridge intersections. It is based on the assumption that the user already has chosen a meaningful time scope for the analysis, i.e., where the lower bound of the time interval already produces comparably sharp ridges. According to Shadden et al. [12], this will lead to FTLE ridges that already approximately satisfy the advection property. Therefore, the advection property of the intersections can be exploited for making the correspondence finding more robust. We then present a technique to quantify the advection property of the intersections and show how it can be used to optimize the FTLE advection time T for obtaining FTLE fields where the ridge intersections satisfy the advection property up to a prescribed average error. Interestingly, we observed for typical discretizations of FTLE sampling grids that the advection property is not a monotonic function of the finite FTLE advection time: we observed local optima. Hence, our findings indicate that once an appropriate interval of possible finite advection times is identified, it is desirable to choose an optimum inside this interval, possibly by our technique.

2.1 FTLE Ridge Intersection

FTLE sampling is a computationally expensive task. One is therefore typically limited to comparably low resolutions of the FTLE sampling grid. Although there are techniques that adapt the sampling grid to the vicinity of LCS [4] and ridges in

general [10], they are, as many adaptive sampling techniques, susceptible to missed features, or would require a priori knowledge about the data. We therefore base our analysis on a regular sampling of the FTLE and use regions of interest if high resolution is required.

The ridge lines in the resulting scalar FTLE fields are extracted according to Eberly's criterion [3]. The subsequent intersection procedure for obtaining the intersections is a trivial problem in the 2D context investigated in this paper. To assert sufficient quality of the geometric intersections, we impose a minimum intersection angle threshold.

2.2 Intersection Tracking

As illustrated in Fig. 1, we extract forward and reverse FTLE fields for both t_0 and $t_0 + \Delta t$. This leads to two sets of intersections, $\mathbf{i}(t_0)$ at time t_0 and $\mathbf{i}(t_0 + \Delta t)$ at $t_0 + \Delta t$. A straightforward approach would use a very small Δt. This would produce almost identical ridges and hence finding correspondences between their intersections would lead to a trivial tracking problem. Further, the limit case $\Delta t \to 0$ would be used for defining the intersection velocity

$$\mathbf{u}_i(t_0 + \Delta t/2) = (\mathbf{i}(t_0 + \Delta t) - \mathbf{i}(t_0))/\Delta t. \tag{2}$$

We have carried out such an analysis using the quad-gyre example, discussed in Sect. 3.1. Unfortunately, it turns out that ridge extraction tends to be only accurate in the order of the cell size of the FTLE sampling grid, and thus, using small Δt leads to poor accuracy of $\mathbf{u}_i$, because $\mathbf{i}(t_0)$ and $\mathbf{i}(t_0 + \Delta t)$ are too close with respect to the resolution of the FTLE sampling grid. Thus, using larger Δt improves accuracy. The time span Δt can be estimated from the average speed $\bar{u}$ of the vector field and the cell size h of the FTLE sampling grid: $\Delta t = ch/\bar{u}$ with a constant $c > 1$. However, in this case, Δt is not small enough to allow the estimation of the intersection velocity by the linearization (2), i.e., the intersection point cannot be assumed to move at constant speed along a straight line at sufficient precision during Δt.

Using comparably large Δt leads to another problem: intersection correspondences are harder to identify because the ridge intersections move over larger distances; the FTLE ridges move and deform, and may even disappear or new ones might originate. However, since the input to our method is a desired parametrization (see Sect. 2) of FTLE and already satisfies the advection property to some extent, we can utilize the advection property to solve the correspondence problem: the intersection points at time t_0 are advected along path lines to time $t_0 + \Delta t$ and these advected points are checked for correspondence with the ridge intersections at $t_0 + \Delta t$ (Fig. 1). To further avoid erroneous correspondences, a threshold limiting discrepancy δ is imposed and the correspondence is identified as the closest remaining candidate with respect to δ. The measure δ reflects the Lagrangian

advection consistency of the FTLE ridge intersections and is denoted as *advection discrepancy* in our technique.

2.3   Measuring Advection Quality

Various approaches are conceivable for the summarization or aggregation of the advection discrepancy δ into a single quality measure for the whole visualization. For basic considerations we determine $\delta_{\min}(T)$, the minimum δ of all intersections at a given value of T. We also compute the average over all intersections $\bar{\delta}(T)$. Of course, many other techniques, e.g., from statistics, can be applied.

Whereas $\delta_{\min}$ tends to show the best case in terms of advection, $\bar{\delta}$ can be used to get an overall picture of the advection quality of the FTLE ridges. All plots are in units of FTLE sampling grid cell size. This way one can easily choose a limit of advection discrepancy in terms of FTLE grid cells and then visually or numerically identify which regions of T satisfy this requirement.

A potential problem are intersections that do not or do only slowly move over time. It is likely that these intersections exhibit small δ and hence exhibit inappropriately high advection quality since they exhibit low discrepancy only due to the fact that they stand still. This can be addressed by an inverse weighting of δ with the length of the corresponding trajectory from Fig. 1. However, it is unlikely that a time-dependent vector field exhibits a zero over extended time. We did not encounter this problem in the context of CFD simulations, although it could appear in vector fields from other domains.

2.4   Finding Locally Optimal Advection Times T

Having now the building blocks at hand, it would be possible to run an optimization process over all feasible grid resolutions, advection times T, starting times t_0, and ridge sharpness thresholds for a given dataset. According to Theorem 4.4 in [12], this would likely result in infinite resolution and infinite advection time T for all t_0. On the other hand, there is a region of interest for all parameters of the analysis (Sect. 2). This is particularly important for the advection time scope T, which depends on the questions of the user and the application (FTLE ridges typically grow with T and can fill the complete domain in, e.g., convective flows).

Therefore, we perform an analysis of the advection discrepancy by uniform sampling of T inside the prescribed advection time interval and provide a plot together with the T producing global minima of the different discrepancy measures $\delta_{\min}$ and $\bar{\delta}$. Finding the global minimum inside the region of interest of T serves here as a straightforward example. Of course, more sophisticated techniques for data analysis can be applied. An important question that can be interactively and visually answered from the plot of the advection discrepancy is the minimum required FTLE

advection time T that makes sure that the advection property is in the average satisfied up to a given tolerance. In our current approach, we still advocate visual inspection of the plots, in particular because they tend to exhibit outliers and are therefore susceptible to errors if simple automatic analysis techniques are applied.

3   Results and Evaluation

In the following, we demonstrate and evaluate our method on three time-dependent 2D examples: the analytic quad-gyre example (Sect. 3.1) and two CFD simulations of buoyant flow (Sections 3.2 and 3.3).

3.1   Quad-Gyre

The double-gyre example was introduced by Shadden et al. [12] to examine FTLE and LCS and to compare them to vector field topology. This dataset consists of two vortical regions separated by a straight separatrix that connects two saddle-type critical points: one temporally oscillating horizontally at the upper edge and the other synchronously oscillating horizontally along the lower edge (Fig. 2, left). This is a prominent example where vector field topology gives a substantially different result from that by FTLE. This dataset is temporarily periodic. To avoid boundary issues, we use a larger range of field, resulting in four gyres. Therefore, we call this example quad-gyre. Using

$$f(x, t) = a(t)x^2 + b(t)x,$$
$$a(t) = \varepsilon \sin(\omega t),$$
$$b(t) = 1 - 2\varepsilon \sin(\omega t), \tag{3}$$
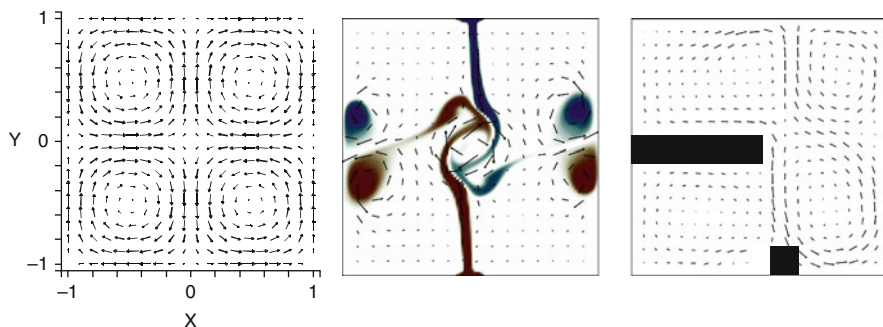
Fig. 2   *Left*: quad-gyre example. *Center*: buoyant plume dataset. *Right*: buoyant flow with obstacles, heated at its lower side and cooled at the top

the quad-gyre is defined in space-time as follows:

$$\mathbf{u}(x, y, t) = \begin{pmatrix} -\pi A \sin(\pi f(x)) \cos(\pi y) \\ \pi A \cos(\pi f(x)) \sin(\pi y) \frac{df}{dx} \\ 1 \end{pmatrix}. \tag{4}$$

As recommended by Shadden et al., we use the configuration $\varepsilon = 1/4$, $\omega = \pi/5$, and $A = 1/10$. Figure 2, left shows a hedgehog plot of the vector field at $t = 0$ and Fig. 3 depicts the FTLE inside a region of interest at the center.

Figure 4 shows a result plot from our method and Fig. 5 a corresponding visualization. It is apparent that the plot of $\delta_{\min}$ is much lower than the plot of $\bar{\delta}$, mostly due to outliers, but both are similar with respect to local extrema and trends.
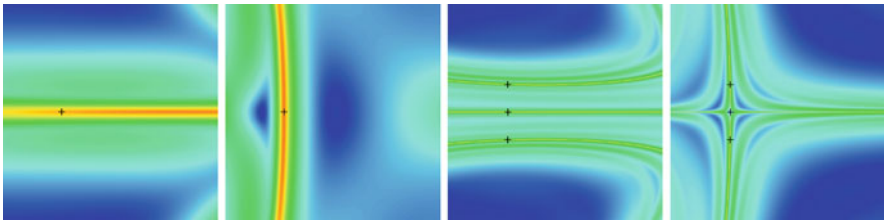
Fig. 3 FTLE visualization for the quad-gyre dataset at time $t_0 = 20$ in a region of interest at the center of the dataset. The FTLE advection times T vary: 4 (*left*), -4 (*left center*), 13 (*right center*), and -13 (*right*). The intersection points between forward-time and reverse-time FTLE ridges are marked by *black crosses*
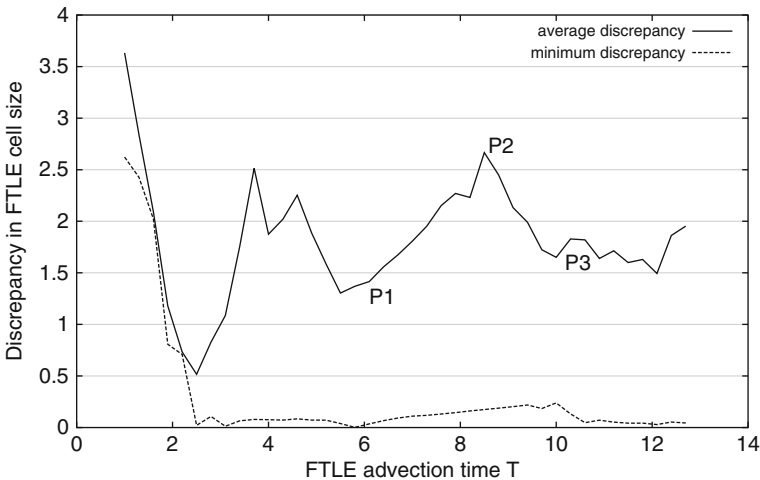
Fig. 4 Quad-gyre example. Discrepancy plots for a setup with FTLE starting time $t_0 = 20$ and ridge intersection advection to time $t = 20.3$ ($\Delta t = 0.3$). Global optimum in $\bar{\delta}$ plot at $T = 2.5$. For three selected values of T, marked as $P1$–$P3$, visualizations are given in Fig. 5
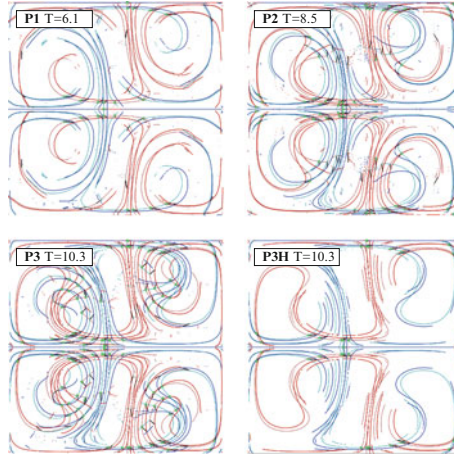
Fig. 5 Quad-gyre example. Forward FTLE ridges in *red* and backward in *blue*; low saturated colors indicate FTLE ridges starting at $t_0 = 20$ and high saturated colors at $t_0 = 20.3$ for three selected values of T marked in the discrepancy plot in Fig. 4. "P3H" is the same as "P3" but with a threshold suppressing ridges that are not sharp. For each intersection at $t_0 = 20$, a pathline is visualized (*black*). If the pathline leads to a corresponding intersection at $t_0 = 20.3$, the starting point is visualized by a *green dot*

It is also apparent how well the ridge sharpness criterion rejected outliers from the analysis. This fact is consistent with the statement by Shadden et al. that an FTLE ridge has to be sufficiently sharp to represent an LCS. Our test directly reflects the fact that as the analysis is restricted to sharp ridges, the advection principle is substantially better satisfied.

We can also well observe from the plot that the advection property is more and more violated as the FTLE advection time T reaches low values. This again supports Shadden et al.'s findings.

3.2 Buoyant Plumes

Our first CFD example is a time-dependent simulation of buoyant plumes inside a 2D box, depicted in Fig. 2, center. The domain is a square of 1 m side length filled with air. The air is initially at rest and at 40 °C. Gravity forces are acting downwards. No-slip conditions are imposed on all boundaries. The left and the right walls are supplied with adiabatic boundary conditions, making the walls neutral in the sense that there is no heat exchange with the outside. There is a region at the center of the lower wall that is heated to 75 °C and a corresponding region on the upper wall that is cooled to 5 °C.

Figure 6 shows the plot from our analysis using $T_0 = 20$ and $\Delta t = 1$. It can be seen that the average discrepancy shows a decreasing trend. We added a plot of
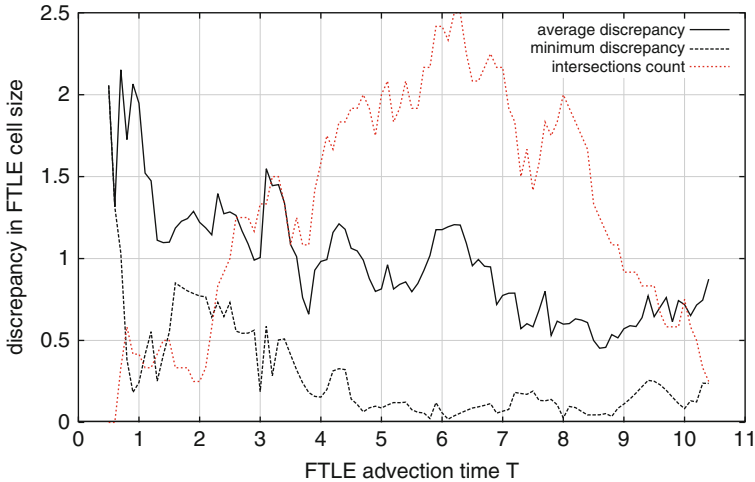
Fig. 6 Buoyant plumes dataset. Examining the plot of $\bar{\delta}$, we identify clearly too low advection times (until $T = 4$), medium quality ranges (around $T = 6$), and comparably high quality (lower than the size of an FTLE sampling cell) for $T = 8$ and larger
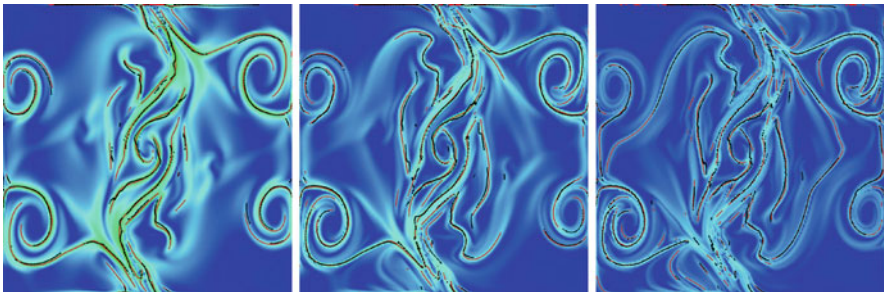
Fig. 7 Buoyant plumes dataset. Advected FTLE ridges (*black dots*) and corresponding ridges (*red lines*) of later FTLE visualizing ridge advection quality, for $T = 4$ (*left*), $T = 6$ (*center*), and $T = 8$ (*right*). The temporal difference Δt between the t_0 of the FTLE fields is 1 s (this is also the time that was used for the advection of the ridges). One can see that with $T = 8$ advection quality is sufficient for typical applications

the number of used intersections for judgment of uncertainty. For validation and for further investigation, we generated images where the complete ridges are advected, not only their intersections (Fig. 7). It is visible in the $\bar{\delta}$ plot that $T = 4$ exhibits high error, $T = 6$ reduced, and $T = 8$ already average error below the size of an FTLE sampling cell. The global optimum inside the $\bar{\delta}$ plot is at $T = 8.6$. The images in Fig. 7 support this finding, it can be seen that for $T = 8$ the advected ridges and the ridges of the corresponding time fit well almost everywhere. It has to be noted that we advected here the repelling ridges, i.e., those from forward FTLE. We also did this for the attracting ridges and there the deviation was much smaller.

Nevertheless, according to the initial motivation of our method, such a comparison with complete advected ridges instead of intersections cannot detect tangential discrepancy between the motion of the ridges and the vector field behavior.

3.3 Buoyant Flow with Obstacle

Our second CFD example is again a buoyant unsteady flow (Fig. 2, right). During the 80 s of simulation time, a convective flow evolves due to the effect of heating the lower wall to 75 °C and cooling down the opposing upper wall to 5 °C. Two rectangular obstacles, a small one on the bottom wall and a larger one on a side wall, prevent the onset of a simple circular flow. We use $t_0 = 10$ to study the early phase of turbulence development (Fig. 2, right). Note that in the following description, time is given in seconds, length units are in meters.

Convective flows are known for their complex topology. We set up an FTLE visualization with $t_0 = 10$ and used a threshold for the eigenvalue of the Hessian to suppress many weak FTLE ridges caused by the turbulent flow. We applied our method for T in the interval $[0.4, 2.6]$, Fig. 8 contains the resulting advection discrepancy plot and Fig. 9 shows visualizations for advection times selected according to the plot. In this analysis, $T = 2.125$ produced the global minimum $\bar{\delta}$ inside the prescribed interval. If advanced data analysis techniques were used, that are able to address noise, it is likely that a lower value would be obtained, more consistent with our observation of a value of $T = 1$. As in our other experiments, the advection discrepancy first decays rapidly as T increases and once
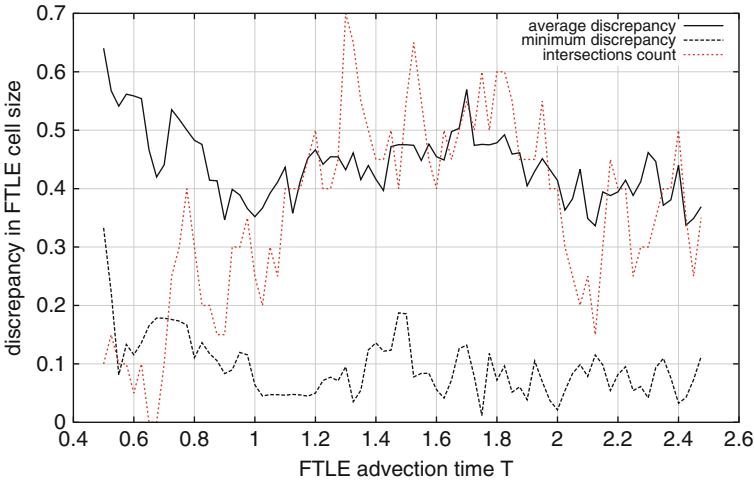
Fig. 8 Buoyancy dataset: average ($\bar{\delta}$) and minimum ($\delta_{\min}$) advection discrepancy plots. Again it is evident that increasing FTLE advection time improves the advection property. Please refer to Fig. 9 for the investigation of selected advection times T ($T = 0.2, 0.5,$ and 1.0)
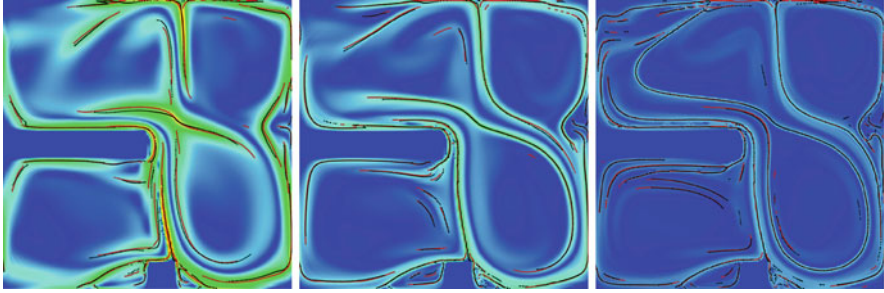
Fig. 9 Buoyancy dataset. Advected FTLE ridges (*black dots*) and corresponding ridges (*red lines*) of later FTLE visualizing ridge advection quality, for $T = 0.2$ (*left*), $T = 0.5$ (*center*), and $T = 1.0$ (*right*). The temporal difference Δt between the t_0 of the FTLE field is 0.05 s (this is also the time that was used for the advection of the ridges). One can see from the $\bar{\delta}$ plot that with $T = 1$ advection quality is getting sufficient for typical applications

it has reached a certain quality, the decay slows down. However, due to the high LCS complexity in this flow, our chosen FTLE sampling grid resolution does not capture the intersections very robustly. This is no surprise since it is known that such flows exhibit very complex, i.e., massively folded LCS, and hence are very difficult to investigate (the ridges in Fig. 9, right are folded, i.e., the line features consist of several ridges).

4 Conclusion

We have presented a technique for measuring the advection property of FTLE ridge intersections in 2D vector fields. Our approach can be seen as dual to the flux-based approach by Shadden et al. By analyzing the temporal behavior of the ridge intersections, not only flow discrepancy orthogonal to the ridges, but also tangential to the ridges can be revealed. Our measurements support the theoretical behavior stated by Theorem 4.4 in [12], i.e., that the error in the advection property tends to decrease with increasing T. We also have observed that the advection property is better satisfied by sharp ridges. An apparent property of our approach is the noise in the order of the size of an FTLE cell. As mentioned, this error seems to be introduced by the ridge extraction stage and hence related to the resolution of the FTLE sampling grid. A thorough analysis is, however, subject to future work. It would also be interesting to compare our approach with the flux-based approach by Shadden et al. and to compare their accuracy with ours. Further, it seems that both approaches could complement each other, possibly leading to a more robust and more accurate technique.

References

1. Asimov, D.: Notes on the topology of vector fields and flows. Technical Report RNR-93-003, NASA Ames Research Center (1993)
2. Benettin, G., Galgani, L., Giorgilli, A., Strelcyn, J.M.: Lyapunov characteristic exponent for smooth dynamical systems and Hamiltonian systems; a method for computing all of them. Mechanica **15**, 9–20 (1980)
3. Eberly, D.: Ridges in Image and Data Analysis. Computational Imaging and Vision. Kluwer Academic Publishers, Dordrecht (1996)
4. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Visual. Comput. Graph. **13**(6), 1464–1471 (2007)
5. Haller, G.: Finding finite-time invariant manifolds in two-dimensional velocity fields. Chaos, **10**(1), 99–108 (2000)
6. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Physica D **149**, 248–277 (2001)
7. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. Computer **22**(8), 27–36 (1989)
8. Kasten, J., Petz, C., Hotz, I., Noack, B., Hege, H.-C.: Localized finite-time Lyapunov exponent for unsteady flow analysis. In: Vision, Modeling, and Visualization, pp. 265–274 (2009)
9. Mathur, M., Haller, G., Peacock, T., Ruppert Felsot, J.E., Swinney, H.L.: Uncovering the Lagrangian skeleton of turbulence. Phys. Rev. Lett. **98**(14) (2007), 144502
10. Sadlo, F., Peikert, R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. IEEE Trans. Visual. Comput. Graph. **13**(5), 1456–1463 (2007)
11. Sadlo, F., Weiskopf, D.: Time-dependent 2-D vector field topology: an approach inspired by Lagrangian coherent structures. Comput. Graph. Forum **29**(1), 88–100 (2010)
12. Shadden, S.C., Lekien, F., Marsden, J.E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. Phys. D: Nonlinear Phenom. **212**(3–4), 271–304 (2005)

Scale-Space Approaches to FTLE Ridges

Raphael Fuchs, Benjamin Schindler, and Ronald Peikert

1 Introduction

The idea behind Lagrangian coherent structures (LCS) is to find material surfaces which separate regions of the fluid with different long term particle movement behavior. In other words, we want to find coherent structures based on their influence on material transport and detect transport barriers in the flow. Attracting LCS are structures on which nearby trajectories accumulate. Conversely, repelling LCS locally exhibit the highest rate of repulsion. The FTLE-based approach to LCS was introduced by the seminal work of Haller and Yuan [12]. Even though FTLE ridges are not always perfect indicators of the LCS [10], they have been shown to work well in a large number of applications [24].

For dynamical systems where there is no limit on integration time and spatial resolution of the field it is possible to use very long integration times and very high sampling densities. For practical data and computational resources this can be impossible to do. Often there is only a finite time interval of the flow available, the spatial resolution of the discretization is limited and the sampling density has to be kept as large as possible to reduce computational effort. In this paper we show that scale-space theory [20] offers answers to these problems.

Originally scale-space was developed for detecting features at different scales in digital images. Typical such features are edges and ridges. There are several benefits of extracting ridges in scale-space:

- When extracting features we have to use operators of finite size for gradient estimation and other operations. The scale-space approach inherently solves the question of which size should be selected.

R. Fuchs (✉) · B. Schindler · R. Peikert
ETH Zurich, 8092 Zurich, Switzerland
e-mail: raphael@inf.ethz.ch; bschindler@inf.ethz.ch; peikert@inf.ethz.ch

- Selecting the scale where the ridges are most distinctive improves the quality of the result.
- The scale of detection contains valuable information about the feature. For example ridges detected only at very small scales of observation are often noise.

Following the related work section, Sect. 3 addresses the numerical computation of FTLE. The two standard ways are described in Sect. 3.1, while Sect. 3.2 explains our scale-space approach. Scale-space extensions to ridge extraction are presented in Sect. 4 and a faster method for computing smoothed flow maps is described in Sect. 5.

2 Related Work

There is a wide range of topological methods for flow visualization. For an overview we refer to state-of-the-art reports on this topic in general [18] and the special case of unsteady flow [25].

Haller [8,9] proposed that a repelling LCS appears as a ridge of the forward-time FTLE field, whereas attracting LCS appear as ridges of the backward-time FTLE field. Shadden et al. [30] define an LCS as a ridge of the FTLE field and derive an estimate for the material flux through FTLE ridges. Tang et al. [31] suggest a method to extend a given finite velocity field to an infinite domain to solve the problem of trajectories leaving the domain. Recently, Haller [11] presented examples of dynamical systems where stable and unstable manifolds cannot be extracted as ridges of the FTLE field and suggested a variational approach to LCS extraction.

The computational effort to compute the flow map and for the extraction of FTLE ridges is substantial. Therefore there are several suggestions on how to reduce the required amount of computation. Garth et al. [6] computed FTLE on planar cross-sections and used direct volume rendering instead of an explicit ridge extraction. In a later work [7], FTLE computation is restricted to offset surfaces at walls. Sadlo et al. [27] and Lekien and Ross [19] suggested an adaptive refinement strategy for ridge extraction that reduces the number of flow map integrations. Sadlo et al. [28], Lipinski and Mohseni [23], and Brunton and Rowley [3] exploit temporal coherency for fast FTLE (ridge) computation for multiple time steps. Jimenez [13] presented a GPU-based FTLE computation for regular 2D grids.

Scale-space theory has been developed by the computer vision community and received much attention in the 1990s. For a general introduction we refer to the book and tutorial of Lindeberg [20, 21]. The main concept is that an image can be extended to a family of smoothed images where the size of the smoothing kernel is parameterized by the scale parameter. Since the detection of features such as ridges is crucially dependent on the size of the gradient estimation stencil, the scale-space representation is a natural approach to select the optimal range for all image operations. Lindeberg [22] discusses a ridge detection approach where an optimal scale is selected at each point based on a strength criterion for a ridge. Florack and

Kuiper [5] propose to analyze the deep structure of an image, that is, to consider all levels of smoothing simultaneously. Scale-space methods are to be distinguished from multi-resolution methods, where data are represented at lower resolutions, e.g. by applying a wavelet transformation, and where typical applications are data compression and progressive visualizations.

In the visualization community there are several works exploring the applicability of scale-space theory. Bauer and Peikert [1] present a technique to extract vortex core lines in scale-space. Klein and Ertl [15] track vector field critical points in scale-space. Kinsner et al. [17] present a GPU implementation for 2D ridge detection. Recently, Kindlmann et al. [16] proposed a scale-space approach to extract crease surfaces in tensor data. An interesting extension to the approach of Lindeberg is that their approach tries to maintain the spatial continuity of the scale.

3   Numerical Computation of FTLE

In a given time-dependent velocity field $\mathbf{u}(\mathbf{x}, t)$ the *trajectory* (pathline) seeded at $(\mathbf{x}_0, t_0)$ is the solution $\phi_{\mathbf{x}_0, t_0}(t)$ of the initial value problem

$$
\begin{aligned}
\frac{\partial}{\partial t}\phi_{\mathbf{x}_0, t_0}(t) &= \mathbf{u}(\phi_{\mathbf{x}_0, t_0}(t), t) \\
\phi_{\mathbf{x}_0, t_0}(t_0) &= \mathbf{x}_0
\end{aligned}
\tag{1}
$$

By keeping t_0 and t fixed, but varying $\mathbf{x}_0$, we obtain the *flow map* $\phi_{t_0}^t(\mathbf{x}_0)$. Its gradient $\mathbf{F} = \nabla \phi_{t_0}^t(\mathbf{x}_0)$ leads to the right Cauchy–Green deformation tensor $\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$ and to the FTLE

$$
\mathrm{FTLE}(\mathbf{x}_0) = \frac{1}{t - t_0} \ln \sqrt{\lambda_{\max}\mathbf{C}}
\tag{2}
$$

3.1   Methods Based on Discretized Flow Map and on Renormalization

Computation of the FTLE field can be done in two ways. In the *flow map method* [9], the flow map is computed on all nodes of a sampling grid, and then the FTLE field is computed using finite differences for the gradients. The *renormalization method* [2] also starts with a finite-difference stencil (given point plus two neighbor points per dimension). The trajectories of the central point and the neighbor points are computed simultaneously, and after each integration step the neighbor trajectory is renormalized, i.e., the distance of the neighbor trajectory and the central trajectory is restored by moving the current point on the straight line connecting the central trajectory and the neighbor trajectory (see Fig. 1). In other words a multiplier is
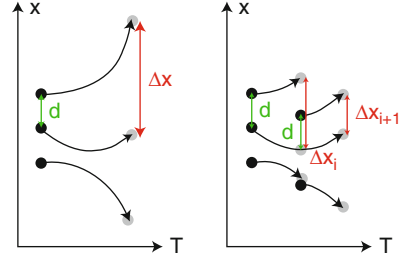
applied to the distance between the central (fiduciary) trajectory and the neighbor trajectory. After integrating over the full time interval $[t_0, t_0 + T]$, the product of these multipliers is computed per trajectory, and the inverse of it is applied to its end point. The difference between the two approaches is that renormalization improves the accuracy of the FTLE at the given point, while the first yields a more representative value for the grid cell represented by the point. In a recent comparison, Kasten et al. [14] used the terms L-FTLE and F-FTLE for FTLE computed with and without renormalization.

By definition, FTLE converge with increasing integration time T to Lyapunov exponents (which exist under the conditions of the Oseledec theorem). This can be used to approximate Lyapunov exponents by computing FTLE with sufficiently large T. However, for this purpose it is crucial to use a method based on renormalization rather than a sampled flow map. The latter approach is easily seen to fail, for example, if the velocity field is fully recirculating, i.e., has no normal component on the domain boundary. In this case the flow map has bounded range, and due to the fixed sampling grid, any estimated flow map gradient is also bounded. Therefore, with growing T, the FTLE estimate converges to zero.

The practical consequence of this is that for FTLE computed with the flow map method, it does not make sense to compare values for different integration times T_1 and T_2, e.g. for numerically checking convergence. Also, concepts such as MFTLE [26] inherit this problem.

Sadlo [26] discusses the problem that FTLE computation with the two methods give quite different results in the case of a flow that splits without shear (see Fig. 2). This case can be illustrated by the velocity field

$$\mathbf{u}(x, y) = \begin{pmatrix} 1 \\ v(x, y) \end{pmatrix} \quad \text{where } v(x, y) = \begin{cases} 0 & x < 0 \\ c\,\mathrm{sgn}(y) & x \geq 0 \end{cases} \tag{3}$$

Its flow map, in the interesting region $-T \leq x \leq 0$, is

$$\phi_{t_0}^{t_0+T}(x, y) = \phi_0^T(x, y) = \begin{pmatrix} x + T \\ y + c(x + T)\mathrm{sgn}(y) \end{pmatrix} \tag{4}$$

and its gradient is

Fig. 2 Synthetic example of a flow splitting at an obstacle at the non-negative *x*-axis
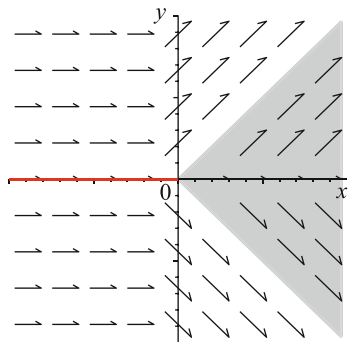
$$\nabla\phi_{t_0}^{t_0+T}(x,y) = \begin{bmatrix} 1 & 0 \\ c\,\mathrm{sgn}(y) & 1 + cT\delta(y) \end{bmatrix} \tag{5}$$

where $\delta(y)$ is the Dirac delta which is infinity at $y = 0$ and zero elsewhere. The FTLE is then also a Dirac delta, for small cT it is approximately $c\delta(y)$.

If the FTLE is computed with renormalization, this works correctly everywhere except on the *x*-axis. However, even though this method is likely to give correct FTLE values everywhere where evaluated, the resulting visualization would miss the only important feature, namely the infinitely thin ridge. For the purpose of visualizing LCS, the renormalization method can therefore not be used. The usual way is to sample FTLE on a grid and to use finite differences for estimating the gradients.

If FTLE is computed with finite differences of the flow map, the infinite gradient at $y = 0$ gets estimated as a value which is inversely proportional to the sampling density. This causes a problem in adaptive methods, where the computed function should not depend directly on the sampling density. The problem is how to define a stopping criterion, given the fact that estimated FTLE values can depend directly on the mesh width.

Even though velocity fields from CFD simulations do not have discontinuities, they have large gradients especially at boundaries. In principle, gradients are bounded, and therefore by sampling the flow map fine enough, the problem could be solved. However, the sampling would have to be at least as fine as the smallest cells of the CFD grid, and even if adaptive sampling was used [27], the refinement cannot be limited to the local density of the CFD grid because the flow map can connect regions of different sampling densities. In practice, this means that the problem of the "flow split without shear" exists, because the necessary sampling density cannot be afforded. The problem can be alleviated by using spline interpolation in the estimation of flow map gradients [6]. The approach presented in the current paper allows to use adaptive sampling of the FTLE field during ridge computation, while avoiding the problem of underestimating the gradient during the process.

3.2 FTLE Computation from Smoothed Flow Map

The proper way of dealing with the above discretization issues is to work in scale-space [20]. Instead of using the original flow map which is possibly sampled too coarsely to resolve all features contained in the underlying numerical velocity field, a smoothed version is used. In the classical scale-space approach, smoothing is done with a Gaussian G_σ with standard deviation σ, which is called the *scale of observation*. The scale of observation can be chosen as fine as the smallest cells of the data grid, or it can be chosen larger to either reduce computation time or focus at larger scale features.

If the flow map from the above example is smoothed with G_σ, then the step function $\text{sgn}(y)$ becomes $\text{erf}(y/(\sqrt{2}\sigma))$ and thus the flow map gradient

$$\nabla \phi_{t_0}^{t_0+T}(x, y) = \begin{bmatrix} 1 & 0 \\ c\,\text{sgn}(y) & 1 + 2cT G_\sigma \end{bmatrix} \tag{6}$$

The FTLE is now (for small cT) approximately $2c\,G_\sigma$. That means, in this example, the FTLE ridge is a Gaussian with a standard deviation that equals the scale of observation. Here, a sampling density of σ would suffice to reconstruct the FTLE ridge.

In general, the relationship between the scale of observation and the minimal sampling density is more complicated, and in particular, depends on the integration time T. It is intuitively clear that the frequencies contained in an FTLE field increase if the integration time T increases, because the longer the integration time is, the longer does the stretching-and-folding effect distort the flow map.

This leads to the assumption that for a given T there is a minimum sampling density for the flow map such that the FTLE can be reliably computed. By convolving this flow map with a Gaussian, high frequencies can be removed again and the LCS (FTLE ridges) are obtained at a larger scale.

We are not aware of any analytical derivation of the minimum sampling density needed to compute the FTLE of a given velocity field for a given T. However, the minimum sampling density can be found algorithmically by performing a (local) subdivision as long as the FTLE value changes more than a prescribed error threshold. When comparing two subdivision levels it is important to do this using the same scale of observation. The effect is that high gradients are no longer estimated inversely proportional to the sampling width. It is also important to compute the flow map gradient not with finite differences but by convolution with derivatives of the Gaussian.

This way, scale-space provides the solution for the adaptive refinement problem. If the scale parameter s is kept fixed, flow map samples can be added adaptively without the aforementioned detrimental effect of the decreasing sampling width. Fig. 3 shows FTLE computed at two different scales from the well-known "double gyre" velocity field [30]. A second example is a simulated 2D air convection flow (Fig. 4).
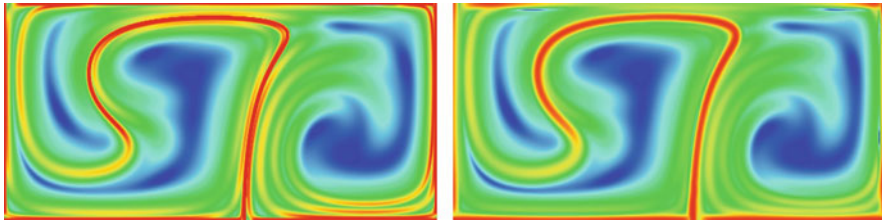
Fig. 3 FTLE of the "double gyre" field, computed with $T = 10$, and colored from 0 (*blue*) to 0.4 (*red*). Scale 0.004 (*left*) and 0.016 (*right*). The size of the domain is 2.0×1.0
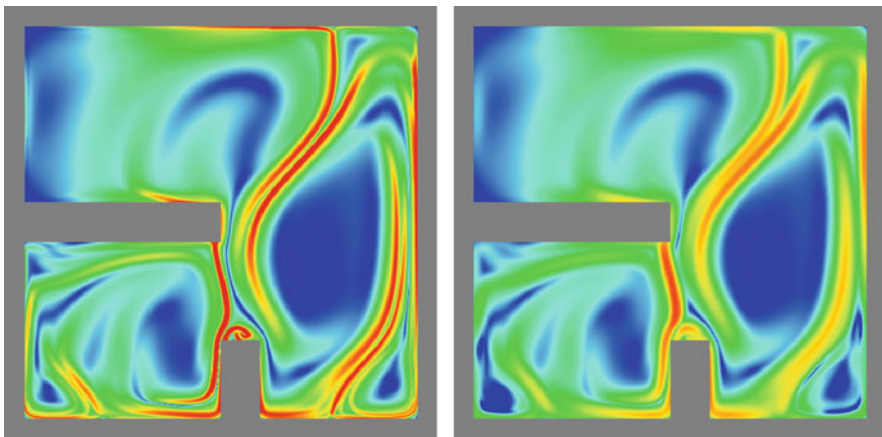
Fig. 4 FTLE of the convection data, computed with $T = 0.5$, and colored from 0 (*blue*) to 7 (*red*). Scale 0.0004 (*left*) and 0.0012 (*right*). The size of the domain is 0.1×0.1

4 Ridge Extraction from FTLE Fields

In this study we did all ridge extraction with the **C**-ridge method [29] which is computationally easier and produces less noisy results than the (maximum convexity) height ridge [4]. According to either ridge definition, a point lies on a co-dimension one ridge if the FTLE field assumes a maximum in a specified transversal direction. The transversal direction is an eigenvector of **C** or of the Hessian of the FTLE, respectively, and the corresponding eigenvalue must be negative and the minimum eigenvalue. Haller postulates [11] that this eigenvector of **C** is perpendicular to the intended ridge direction. Our choice of the **C**-ridge is not crucial, the height ridge could as well be chosen. For extracting FTLE ridges we need as precomputed data the FTLE value and the transversal direction on the points of a (regular or adaptive) sampling grid. The ridge extraction is then done per sampling point and consists in finding a nearby FTLE maximum along the transversal direction, where "nearby" means within a one-cell neighborhood. Finding the FTLE maximum is done using bisection and by looking for a zero
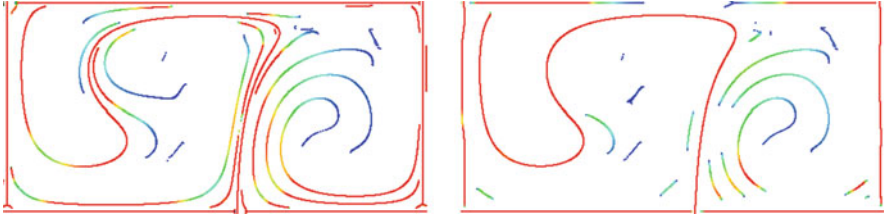
Fig. 5 Ridges of the FTLE fields from Fig. 3, colored by ridge strength (*red*=high). At a higher scale many of the smaller ridges are removed
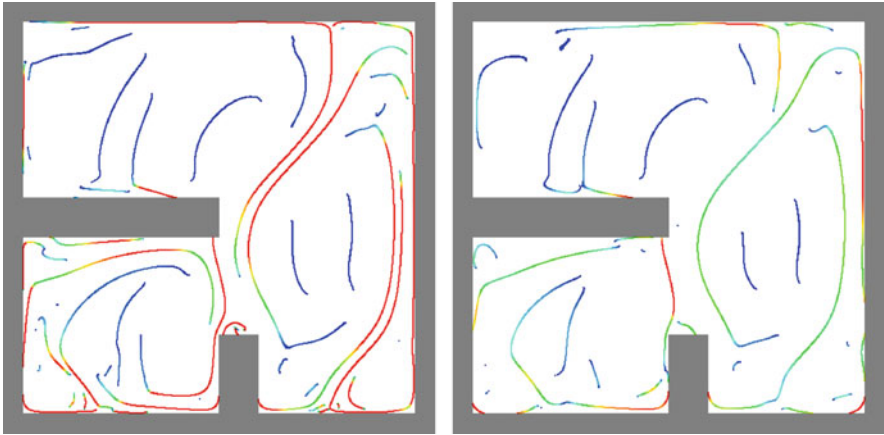
Fig. 6 Ridges of the FTLE fields from Fig. 4, colored by ridge strength (*red*=high). At a higher scale, we can see that the ridge separating the large and the small room is the most prominent

crossing of the FTLE gradient. The FTLE gradient is estimated by convolution with the gradient of a Gaussian, the standard deviation of which is the scale parameter of the ridge extraction (Figs. 5 and 6).

4.1 Optimal Scale Ridges of FTLE

The FTLE field can be interpreted as a 2D or 3D image, and therefore the concept of scale-space ridges [22] is directly applicable. It will extract ridges of different scales by automatically adapting the local scale of observation to the "width" of the ridge. A *scale-space ridge point* is a ridge point with the additional property that it is a maximum of the *ridge strength* in the transversal direction given by the particular ridge definition. Among the various measures for ridge strength, we chose the *normalized curvature*, i.e., the second derivative of FTLE in the transversal direction, multiplied with the normalization factor $s^{3/4}$, where s is the scale parameter.
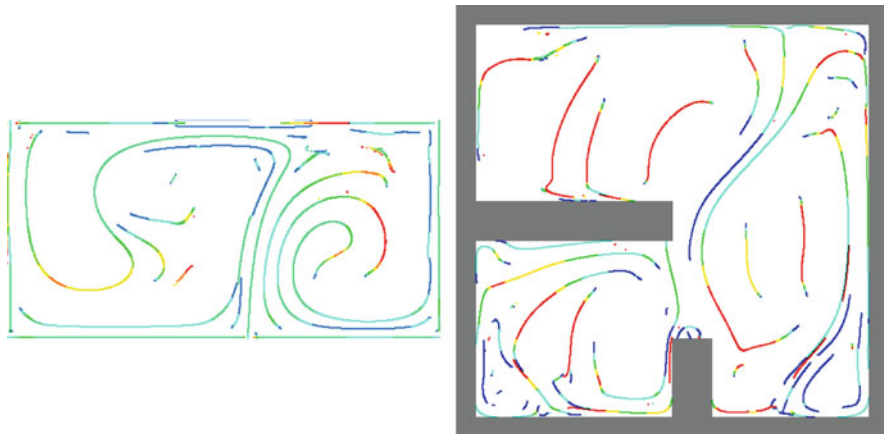
Fig. 7 *Left*: Scale-space ridges of FTLE computed with $\sigma = 0.08$ (double gyre) and $\sigma = 0.004$ (convection flow). Ridges are colored by scale. *Left*: double gyre, *blue*=0.25σ, *red*=2σ. *Right*: convection, *blue*=0.5σ, *red*=1.5σ

The procedure for extracting scale-space extends the one described in Sect. 4 by a loop over the scales. In addition, any ridge point obtained at scale s_i is checked for being also a scale-space ridge point. For this, corresponding ridge points in scales s_j (with $j = i-1, i, i+1$) are searched for on the line given by the transversal direction. Then the ridge strength for the three ridge points is computed. The required second derivative of the FTLE in the transversal direction is computed using the Hessian of FTLE which is obtained by convolution with the Hessian of G_{s_j}.

When extracting scale-space ridges from our two test data (Fig. 7) it turned out that ridge points were indeed extracted at different scales. However, by comparing the scale-space ridges with a set of fixed-scale ridges (using the same scale for FTLE computation and for ridge extraction), it turned out that the scale-space ridge is very similar to one of the fixed-scale ridges. Surprisingly, however, this fixed scale turned out to be smaller than the σ that was used in the FTLE computation. In both cases, this scale is roughly 0.75σ. We do not claim that there exists such a factor, but it seems that it is not optimal to extract ridges at the same scale as was used for FTLE computation. A slightly smaller scale may yield more meaningful detail.

There exists an alternative way of computing scale-space ridges of FTLE. Rather than using a fixed FTLE "image", FTLE computation can be done on-the-fly and at the same scale as the subsequent ridge extraction. The results (Fig. 8) again are visually close to ridges extracted at a single scale. In Fig. 8, left the range of scales was chosen such that even the smallest scale produces a smoothed FTLE image. Consequently, the scale-space ridges are similar to the fixed-scale ridges at this lowest scale. Larger scales do not contribute much besides slightly deforming the ridges. In Fig. 8, right the flow map has only large-scale features, and the range of
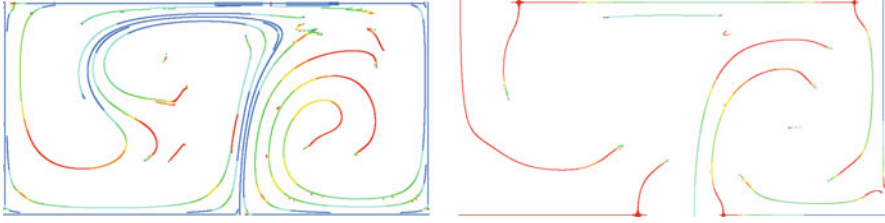
Fig. 8 Scale-space ridges where computation of FTLE is done on-the-fly for the double gyre data. Integration time was $T = 10$ (*left*) and $T = 4$ (*right*). Range of scales: 0.002 (*blue*) to 0.016 (*red*)

scales was chosen such that it extends below the minimum scale of the features. Here, the ridge image is similar to a fixed-scale image at an intermediate scale.

Based on the observation that larger scales have little contribution to FTLE ridges, the only potential advantage of the scale-space ridge method is that it can detect the minimum feature scale. However, this information is also obtained from adaptive FTLE computation. It is certainly more efficient to first compute an FTLE and let it follow by a ridge extraction using the same scale, or possible a slightly smaller scale, taking the above observation into account. This can be done also if a spatially varying scale is used.

4.2 Optimal Integration Time Ridges of FTLE

In Sect. 4.1 it was assumed that the integration time T was given and an optimal scale was searched for. However, in the context of FTLE ridges it is even more useful to search for an optimal integration time, given a scale of observation. Computing such optimal integration time ridges is similar to computing scale-space ridges (Sect. 4.1), with looping over integration times instead of scales. However, when comparing ridge strengths for two different integration times, we have to account for the fact that ridges tend to get stronger with increasing T. The deformation of a "fluid" element under the flow map is expressed by the Cauchy-Green tensor $\mathbf{C}$. The square root of the smallest eigenvalue of this tensor, $\lambda_{\min}\mathbf{C}$, expresses contraction in the direction where this is maximized. This direction tends to be aligned with the FTLE ridges, therefore we propose to modify ridge strength by multiplying it with $\lambda_{\min}\mathbf{C}$.

Such scale-space ridges based on optimal integration time are shown in Fig. 9. When looking at the strong ridge originating near the center of the bottom line, it can be seen that this feature is represented twice. A short-time version of it is curved to the right, while a long-time version is curved to the left. They both represent approximations for material structures, having different life-times. Any ridge extraction at a fixed T would give at most one of these two manifestations of the ridge.
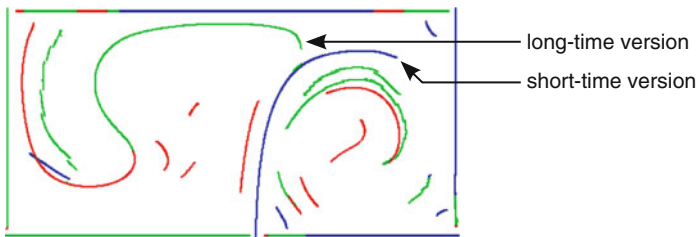
Fig. 9 Ridge of double gyre at multiple integration times T ranging from 4 (*blue*) to 10 (*red*). We can see how the main separation in the double gyre is detected better as a combination of a long-term FTLE ridge and a short term FTLE ridge

5 Incremental Computation of the Flow Map

If FTLE are to be computed for multiple initial times or multiple integration times, the reuse of computed trajectories can save computing time. In the case of a fixed initial time t_0, this is easily done by extending trajectories at their end points. If trajectories are seeded on a regular grid, this results in all flow maps sampled on this regular grid. The case of fixed T but variable t_0, needed for animated visualization, is more difficult, because shortening trajectories at their beginning results in flow maps sampled on distorted grids. This approach is possible [28], but requires frequent re-initialization to avoid overly distorted grids.

A better alternative is to compute partial flow maps for a sequence of initial times $t_0 + i\Delta t$ and integration time Δt. A contiguous subset of these are then composited to the flow map for the desired sliding time window. Composition of flow maps requires their values at arbitrary points, which are obtained by convolution with a Gaussian kernel. Of course, such a repeated smoothing requires the choice of a smaller σ. Convolution with G_σ repeated N times is equivalent to convolution with $G_{\sqrt{N}\sigma}$. In the proposed incremental approach convolution alternates with application of partial flow maps. Therefore, it is hard to determine the standard deviation for the single flow map method that exactly corresponds to the partial flow map method with G_σ. Nevertheless, our experiments showed that $\sqrt{N}\sigma$ works in a qualitative sense.

With this setting, the partial flow map method produced FTLE values with a mean absolute error of 0.025, at a data range of $[-1.16, 7.36]$, which are barely noticeable in Fig. 10. In the extracted ridges, the error is mostly below the sampling width of the grid (Fig. 10, right). In the "double gyre" example, the mean absolute error is only 0.0004, at a data range of $[0.005, 0.424]$ and the extracted ridges are visually indistinguishable.

Often, an exact scale value is not required, e.g., when computing optimal-scale ridges (Sect. 4.1). It has to be pointed out that the number of partial flow maps must be kept relatively small because the standard deviation cannot be chosen arbitrarily small. We observed that for sufficient approximation quality this should be about 2.0 times the sampling width.
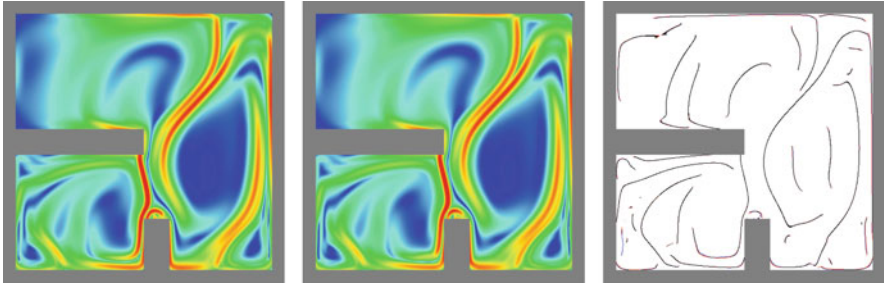
Fig. 10 (*Left*) FTLE computed with single flow map and (*middle*) composition of partial flow maps, colored from 0 (*blue*) to 7 (*red*). The faster, approximate method (*middle*) yields visually almost identical results. (*Right*) Ridges (at scale 0.0008) of the above two FTLE fields computed with single (*red*) and partial (*blue*) flow map method, colored *black* where identical within pixel precision

The incremental computation reduces the integration time by a factor $1/N$, assuming that a sequence of flow maps for successive times is computed. Gaussian smoothing has to be done N times instead of just once. If done efficiently, computing time for one such smoothing is dominated by the forward and inverse FFT operation. This means that a speedup of close to N can be achieved by using the partial flow map method.

6   Conclusion

In this paper we showed that the scale-space offers an elegant way to cope with undersampling problems that can occur due to limited space and computing time availability. This way, smoothed FTLE fields can be correctly computed, even in adaptive methods, where existing methods produced results that were biased by the sampling width. Also, a series of smoothed FTLE in a sliding time window can be computed efficiently using a set of partial flow maps.

An often criticized problem of FTLE is its dependence on a parameter, the integration time. While FTLE itself is not parameter-free, FTLE ridges become parameter-free with our proposed definition of optimal-integration time ridges.

Finally, a topic that deserves further study is how to handle trajectories that prematurely leave the domain. Given the strong dependence of FTLE on the choice of T, it seems problematic to include such trajectories into FTLE visualizations and into further processing such as ridge extraction. On the other hand, excluding them means to exclude large parts of the domain from visualization. Fully recirculating flow, as we used in our examples, is the exception rather than the rule in application-related velocity fields.

References

1. Bauer, D., Peikert, R.: Vortex tracking in scale space. In: Ebert, D., Brunet, P., Navazo, I. (eds.) Data Visualization 2002: Proceedings of the 4th Joint EUROGRAPHICS—IEEE TCVG Symposium on Visualization (VisSym 2002), Eurographics, pp. 233–240 (2002)
2. Benettin, G., Galgani, L., Giorgilli, A., Strelcyn, J.-M.: Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them. Part 1: theory. Meccanica **15**, 9–20 (1980)
3. Brunton, S.L., Rowley, C.W.: Fast computation of finite-time Lyapunov exponent fields for unsteady flows. Chaos **20**(1), 017503.1–017503.9 (2010)
4. Eberly, D.: Ridges in Image and Data Analysis. Computational Imaging and Vision. Kluwer Academic Publishers, Dordrecht (1996)
5. Florack, L., Kuijper, A.: The topological structure of scale-space images. J. Math. Imag. Vis. **11**(1), 365–79 (2000)
6. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Visual. Comput. Graph. **13**(6), 1464–1471 (2007)
7. Garth, C., Wiebel, A., Tricoche, X., Joy, K., Scheuermann, G.: Lagrangian visualization of flow-embedded surface structures. Comput. Graph. Forum **27**(3), 1007–1014 (2008)
8. Haller, G.: Finding finite-time invariant manifolds in two-dimensional velocity fields. Chaos **10**(1), 99–108 (2000)
9. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Physica D **149**, 248–277 (2001)
10. Haller, G.: Lagrangian coherent structures from approximate velocity data. Physics of Fluids **14**, 1851–1861 (2002)
11. Haller, G.: A variational theory of hyperbolic Lagrangian coherent structures. Phys. D: Nonlinear Phenom. **240**(7), 574–598 (2010)
12. Haller, G., Yuan, G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. Physica D **147**(3), 352–370 (2000)
13. Jimenez, R., Vankerschaver, J.: Optimization of FTLE Calculations Using nVidia's CUDA. Technical Report, California Institute of Technology (2009)
14. Kasten, J., Petz, C., Hotz, I., Noack, B., Hege, H.-C.: Localized finite-time lyapunov exponent for unsteady flow analysis. In: Magnor, M., Rosenhahn, B., Theisel, H. (eds.) Vision Modeling and Visualization, vol. 1, pp. 265–274, Universität Magdeburg, Inst. f. Simulation u. Graph. (2009)
15. Klein, T., Ertl, T.: Scale-space tracking of critical points in 3D vector fields. In: Hauser, H., Hagen, H., Theisel, H. (eds.) Topology-Based Methods in Visualization, pp. 35–49. Springer, Heidelberg (2007)
16. Kindlmann, G.L., San Jose Estepar, R., Smith, S.M., Westin, C.-F.: Sampling and visualizing creases with scale-space particles. IEEE Trans. Visual. Comput. Graph. **15**(6), 1415–1424 (2009)
17. Kinsner, M., Capson, D., Spence, A.: Scale-space ridge detection with GPU acceleration. In: Canadian Conference on Electrical and Computer Engineering, 2008. CCECE 2008, Niagara Falls, pp. 1527–1530 (2008)
18. Laramee, R., Hauser, H., Zhao, L., Post, F.: Topology-based flow visualization, the state of the art. In: Hauser, H., Hagen, H., Theisel, H. (eds.) Topology-Based Methods in Visualization, pp. 1–20. Springer, Heidelberg (2007)

19. Lekien, F., Ross, S.D.: The computation of finite-time Lyapunov exponents on unstructured meshes and for non-Euclidean manifolds. Chaos **20**(1), 017505.1–017505.20 (2010)
20. Lindeberg, T.: Scale-Space Theory in Computer Vision. The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, Dordrecht (1994)
21. Lindeberg, T.: Scale-space: a framework for handling image structures at multiple scales. In: Proceedings of CERN School of Computing (1996).
22. Lindeberg T.: Edge detection and ridge detection with automatic scale selection. Int. J. Comput. Vis. **30**(2), 117–154 (1998)
23. Lipinski, D., Mohseni, K.: A ridge tracking algorithm and error estimate for efficient computation of Lagrangian coherent structures. Chaos **20**(1), 017504.1–017504.9 (2010)
24. Peacock, T., Dabiri, J.: Introduction to focus issue: Lagrangian coherent structures. Chaos **20**(1), 017501.1–017501.3 (2010)
25. Pobitzer, A., Peikert, R., Fuchs, R., Schindler, B., Kuhn, A., Theisel, H., Matkovic, K., Hauser, H.: On the way towards topology-based visualization of unsteady flow—the state of the art. In: Hauser, H., Reinhard, E. (eds.) State of the Art Reports, Eurographics Association, pp. 137–154 (2010)
26. Sadlo, F.: Computational Visualization of Physics and Topology in Unsteady Flow. Ph.D. Dissertation No. 19284, ETH Zurich (2010)
27. Sadlo, F., Peikert, R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. IEEE Trans. Visual. Comput. Graph. **13**(6), 1456–1463 (2007)
28. Sadlo, F., Rigazzi, A., Peikert, R.: Time-dependent visualization of Lagrangian coherent structures by grid advection. In: Pascucci, V., Tricoche, X., Hagen, H., Tierny, J. (eds.) Topological Data Analysis and Visualization: Theory, Algorithms and Applications, pp. 151–166. Springer, Heidelberg (2010)
29. Schindler, B., Peikert, R., Fuchs, R., Theisel, H.: Ridge concepts for the visualization of Lagrangian coherent structures. In: Peikert, R., Hauser, H., Carr, H., Fuchs, R. (eds.) Topological Methods in Data Analysis and Visualization II, Mathematics and Visualization, pp. 221-236, Springer, Heidelberg (2012)
30. Shadden, S., Lekien, F., Marsden, J.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. Phys. D: Nonlinear Phenom. **212**, 271–304 (2005)
31. Tang, W., Chan, P. W., Haller, G.: Accurate extraction of LCS over finite domains, with application to flight safety analysis over Hong Kong International Airport. Chaos **20**(1), 017502.1–017502.8 (2010)

Index