DISSERTATION

submitted to the

COMBINED FACULTY FOR THE
NATURAL SCIENCES AND MATHEMATICS

of

HEIDELBERG UNIVERSITY, GERMANY

for the degree of

Doctor of Natural Sciences

put forward by

Lutz Hofmann, M.Sc.

born in Freiburg i. Br., Germany

Heidelberg
September 2021

Local Features and Topology in Higher-Dimensional and Time-Dependent Vector Fields

by

LUTZ HOFMANN

Advisor: Prof. Dr. Filip Sadlo

Oral examination: ____________________

Abstract

Vector fields can be used to describe a wide variety of physical phenomena. Their structure can prominently be understood by appropriate feature extraction and visualization. While existing methods mainly treat 2D and 3D vector fields, this thesis focuses on higher-dimensional vector fields of dimension four and beyond, as well as the special case of treating time as additional dimension.

In the first part of this thesis, we introduce the dependent vectors operator, which enables to define and extract a number of local features in scalar and vector fields of arbitrary dimension. It generalizes the 3D parallel vectors operator by replacing the parallelism condition with linear dependency of a set of (possibly derived) vector fields. The resulting manifolds, whose dimension depends on the number of vector fields considered, are obtained by a generic extraction algorithm. We demonstrate their utility by introducing generalized vortex core manifolds, bifurcation manifolds, as well as ridge manifolds, which we extract from dynamical systems as well as higher-dimensional fields derived from 2D and 3D numerical datasets.

Vector field topology (VFT), a global feature describing qualitative transport structure, is treated in the second part of this thesis. First, we generalize the well-established 2D and 3D VFT to four-dimensional steady vector fields. We classify the different types of 4D critical points, propose a set of glyphs made of 4D geometry to represent each type, extract and classify 4D periodic orbits, and extract their invariant manifolds. For effective exploration of the resulting 4D topological skeleton, we introduce a 4D camera that exploits degeneracies in the 3D projections to reduce clutter and self-intersection. We exemplify the utility of our approach using analytic 4D dynamical systems that showcase properties of 4D VFT.

Finally, we make several contributions to 2D and 3D time-dependent vector field topology. In this concept, the role of invariant manifolds from steady VFT is taken on by Lagrangian coherent structures (LCS), which represent the main organizing streak manifolds over finite time intervals. Instead of the commonly employed expensive computation of the LCS by ridge extraction from the finite-time Lyapunov exponent (FTLE) field, our approach is based on the refinement of locally extracted candidate manifolds. For the 2D case, we combine and extend existing concepts for obtaining initial candidates and refinement toward hyperbolic trajectories, and show that our approach is more accurate and efficient than existing methods. In the 3D case, we present a novel method for obtaining hyperbolic path surfaces from candidate surfaces, which we extract locally from the four-dimensional space-time domain. We evaluate our approach

on analytic flows, as well as data from computational fluid dynamics, using the FTLE as a ground truth superset.

Zusammenfassung

Vektorfelder können zur Beschreibung einer Vielzahl von physikalischen Phänomenen verwendet werden. Ihre Struktur lässt sich insbesondere durch geeignete Methoden der Merkmalsextraktion und Visualisierung gut verstehen. Während bestehende Methoden hauptsächlich 2D und 3D Vektorfelder behandeln, konzentriert sich diese Arbeit auf höherdimensionale Vektorfelder der Dimension vier und höher, sowie auf den Spezialfall, Zeit als zusätzliche räumliche Dimension zu behandeln.

Im ersten Teil dieser Arbeit wird der Dependent-Vectors-Operator eingeführt, der es ermöglicht, eine Reihe von lokalen Merkmalen in Vektorfeldern beliebiger Dimension zu definieren und zu extrahieren. Dieser Operator verallgemeinert den 3D Parallel-Vectors-Operator, indem die Parallelitätsbedingung durch lineare Abhängigkeit einer Menge von (möglicherweise abgeleiteten) Vektorfeldern ersetzt wird. Die resultierenden Mannigfaltigkeiten, deren Dimension von der Anzahl der betrachteten Vektorfeldern abhängt, berechnen wir mit einem generischen Extraktionsalgorithmus. Wir demonstrieren ihren Nutzen durch die Definition verallgemeinerter Vortex-Core-Mannigfaltigkeiten, Bifurcation-Mannigfaltigkeiten sowie Ridge-Mannigfaltigkeiten, die wir aus dynamischen Systemen, sowie aus höherdimensionalen Feldern, die aus numerischen 2D und 3D Datensätzen abgeleitet wurden, extrahieren.

Vektorfeldtopologie (VFT), ein globales Merkmal, das die qualitative Transportstruktur beschreibt, wird im zweiten Teil dieser Arbeit behandelt. Zunächst verallgemeinern wir die etablierte 2D und 3D VFT auf vierdimensionale stationäre Vektorfelder. Wir klassifizieren die verschiedenen Typen von 4D kritischen Punkten, definieren eine Reihe von Glyphen bestehend aus 4D Geometrie, um die jeweiligen Typen zu repräsentieren, extrahieren und klassifizieren 4D periodische Orbiten, und extrahieren die zugehörigen invarianten Mannigfaltigkeiten. Zur effektiven Erkundung des resultierenden topologischen 4D Skeletts führen wir eine 4D Kamera ein, die Entartungen in den 3D Projektionen ausnutzt, um visuelle Überladung und Selbstüberschneidungen zu reduzieren. Wir veranschaulichen die Anwendbarkeit unseres Ansatzes anhand von 4D dynamischen Systemen, welche die besonderen Eigenschaften der 4D VFT aufzeigen.

Schließlich leisten wir mehrere Beiträge zur 2D und 3D zeitabhängigen Vektorfeldtopologie. In diesem Konzept wird die Rolle der invarianten Mannigfaltigkeiten aus der stationären VFT von Lagrangian Coherent Structures (LCS) übernommen, die die wichtigsten organisierenden Streichmannigfaltigkeiten über endliche Zeitintervalle darstellen. Anstelle der üblicherweise verwendeten teuren Berechnung der LCS durch Ridge-Extraktion aus dem Finite-Time-Lyapunov-Exponent-Feld (FTLE) basiert unser Ansatz auf der Verfeinerung von lokal extrahierten Kandidaten-Mannigfaltigkeiten. Für den 2D-Fall kombinieren und erweitern wir bestehende Konzepte zur Extraktion von initialen Kandidaten und der Verfeinerung in Richtung hyperbolischer Trajektorien und zeigen, dass unser Ansatz genauer und effizienter ist als bestehende Methoden. Im 3D-Fall stellen wir eine neuartige Methode vor, um hyperbolische Pfadflächen aus Kandidatenflächen zu erhalten, die wir lokal aus dem vierdimensionalen Raum-Zeit-Vektorfeld extrahieren. Wir evaluieren unseren Ansatz an analytischen Strömungen sowie an Daten aus der rechnergestützten Strömungsmechanik, wobei wir FTLE als Ground-Truth-Obermenge verwenden.

Acknowledgements

Many people have contributed to the successful completion of this work. This section is dedicated to them.

First of all, I would like to thank my advisor Filip Sadlo for his guidance and support. Without his enthusiasm and passion for teaching and sharing his knowledge, I would not have considered to pursue a PhD. His tireless strive for perfection and attention to detail is what made our initial ideas evolve into research projects. I further thank Peter Albers and Susanne Krömker for their patience for explaining and discussing mathematical concepts. I would like to thank Bastian Rieck for sharing his seemingly endless wisdom about academia and life as doctoral student, most of which has since proven true. I am grateful for my former office mates Philipp Jung and Karsten Hanser, and our many cups of afternoon tea, which not only were some of my favorite times of the day, but also led to some of the most productive discussions about our research. Furthermore, I thank Heiko Augustin for our almost weekly interdisciplinary breakfast meetings, which allowed me to see my work from new perspectives.

I would like to thank Julien Tierny and the TTK team for allowing me to participate in their open source project. This work has helped me regain motivation for my own work, and provided much inspiration for keeping my own code base organized.

The many motivated students/inhabitants of the Graphics Lab made me look forward to being at the office as early as possible every day. I especially thank Conrad Sachweh and Felix Feldmann for their invaluable help with Linux system administration, as well as the many shared glasses of gin and tonic on the rooftop terrace. The latter certainly made working long hours much more bearable.

This work rests on the foundations acquired during my undergraduate studies, which would not have been successful without the mutual help of my fellow students and friends, Viola Caspari, Mirko Link, Claudia Ridinger, Sebastian Nill, Yu E Tay, Moritz Gomm, Felix Rumpel, Elli Dunayevska, and many others.

Finally, I thank my wife Yung-Hsin Shih for her emotional support and for pushing me to work even harder.

Contents

1 Introduction

Many physical phenomena in our everyday life, such as those involving force or motion, can be described by 2D or 3D time-independent vector fields. However, there are phenomena that can only be appropriately modeled using higher-dimensional vector fields. For example, treating time as additional dimension also leads to higher-dimensional space-time vector fields. On the other hand, the motion of inertial objects due to forces in 2D or 3D space is described by respectively 4D or 6D vector fields, called the phase space. Considering time as additional dimension in the phase space leads to 5D and 7D vector fields, respectively. In general, any phenomenon that is described by a continuous dynamical system with n variables has an underlying n-dimensional vector field. Beyond that, higher-dimensional scalar fields can arise from families of 2D or 3D vector fields that depend on one or more additional parameters. For example, scalar fields that are derived from the flow map, which maps the position of particles seeded at an initial time to their position after a certain advection time, add initial time and advection time to the spatial dimensions, and are thus 4D or 5D in that context.

In real-world experiments, the flow of fluids can usually be understood when made visible by injecting a tracer such as ink or smoke, i.e., it needs to be *visualized* experimentally. Dynamical systems given as analytic formulas or data resulting from computational fluid dynamics share this property. Computational *flow visualization* is the study of algorithms and methods for the visualization of such data. Besides direct visualization approaches, such as arrow glyphs and color mapping, feature extraction has proven effective for reducing the data to their essential structure. The structures extracted from vector fields are, however, typically only meaningful by their relative position within the phase space. In two- and three-dimensional vector fields, 2D and 3D rendering techniques from computer graphics can be readily employed in such cases and yield intuitive representations.

The focus of this thesis are four- and higher-dimensional vector fields, as well as 3D time-dependent vector fields, which possess four-dimensional space-time domains. Due to their high dimensionality, such fields and the structures contained in them need to be projected onto a 2D or 3D canvas in order to display them. Therefore, to avoid clutter and self-intersection in such projections, the extraction of suitable features of lower dimensionality that represent meaningful structures within such fields is highly useful. The definition and extraction of features in higher-dimensional (vector) fields has, however, not yet received sufficient treatment in existing visualization literature.

1.1 Contributions

This thesis is split into two parts. In the first part, we consider local features in higher-dimensional (derived) vector fields, while the second part deals with a particular non-local feature, vector field topology, as well as its time-dependent generalization.

We first consider the extraction of locally defined features from vector fields of arbitrary dimension. For 3D vector fields, the parallel vectors (PV) operator has proven effective for the definition of line-type features such as vortex core lines, which represent curves around which particles exhibit swirling motion. We introduce the dependent vectors (DV) operator [HS19], which generalizes the PV operator to arbitrary dimension of the vector fields as well as arbitrary dimensionality of the features. Hereby, the parallelism condition of two vector fields is replaced by the linear dependence of a set of vector fields. We propose a generic algorithm for extracting the manifolds defined by our generalization, and apply our approach for the generalization of vortex core manifolds, bifurcation manifolds, and ridge manifolds.

We then turn to the visualization of 4D vector field topology [HRS18]. To this end, we classify the types of four-dimensional critical points and construct a set of glyphs depicting them in terms of four-dimensional geometry. We present a projection-based visual representation of (glyph) geometry, and manifold-based exploration of the four-dimensional topological skeleton. Additionally, we classify the different types of four-dimensional periodic orbits, and construct example vector fields for each case.

Finally, we present a local extraction approach for 2D and 3D time-dependent vector field topology [HS20; HS21]. In the 2D case, we combine existing local extraction techniques for hyperbolic trajectories with global numerical approaches and refinement. Thereby, we obtain consistent seeding directions for the associated streak manifolds, which we enhance with an automatic selection algorithm for the required seeding lengths. This yields a more accurate and reliable extraction of 2D time-dependent vector field topology than with previous methods. For the 3D case, we introduce a local extraction method for candidate surfaces in 4D space-time, which can conceptually be defined using our dependent vectors operator, and a robust and accurate refinement toward the aimed hyperbolic path surfaces. We demonstrate the necessity to additionally include unsteady equivalents of bifurcation lines, spiral-saddle critical points, as well as a class of saddle-type periodic orbits. Finally, we evaluate different approaches to obtain initial candidates, and compare our technique with previous approaches. As ground truth, we use the finite-time Lyapunov exponent (FTLE), which measures exponential separation of infinitesimally close particles. Since in the 3D case, the space-time vector field is four-dimensional, insights from our visualization of steady 4D vector field topology into saddle connectors can be transferred here.

1.2 Structure of this Thesis

In Chapter 2, we introduce the notions used in this thesis, and provide an overview over existing literature in relation to this work. Chapter 3 presents a generalization of the parallel vectors operator to arbitrary dimension and number of linearly dependent vector fields. In Chapter 4, we propose an approach to the visualization of the topology of steady four-dimensional vector fields. Chapter 5 introduces the concept of distinguished hyperbolic trajectories for the extraction of 2D time-dependent vector field topology, which we further extend to the 3D case. This thesis is concluded in Chapter 6, where we also point out potentials for future research.

1.3 Publications

The work presented in this thesis is based on the following publications in the peer-reviewed journal *Computer Graphics Forum*:

- L. Hofmann, B. Rieck, and F. Sadlo. “Visualization of 4D vector field topology”. *Computer Graphics Forum* 37:3, 2018, pp. 301–313.
- L. Hofmann and F. Sadlo. “The dependent vectors operator”. *Computer Graphics Forum* 38:3, 2019, pp. 261–272.
- L. Hofmann and F. Sadlo. “Extraction of distinguished hyperbolic trajectories for 2D time-dependent vector field topology”. *Computer Graphics Forum* 39:3, 2020, pp. 303–315.
- L. Hofmann and F. Sadlo. “Local extraction of 3D time-dependent vector field topology”. *Computer Graphics Forum* 40:3, 2021, pp. 111–122.

1.4 Notation

We use the following mathematical notation throughout this thesis:

c	Scalars
$f(\mathbf{x})$	Scalar fields
$\mathcal{S}$	Sets
$\mathbf{x}, x_i$	Column vectors with components $x_1, \dots, x_n$
$\mathbf{A}$	Square matrices
$\det(\mathbf{A})$	Determinant of a square matrix $\mathbf{A}$
$\det(\mathbf{v}_1, \dots, \mathbf{v}_n)$	Determinant of the matrix with columns $\mathbf{v}_1, \dots, \mathbf{v}_n$
$\boldsymbol{\zeta}$	Local coordinates
$\mathbf{u}(\mathbf{x})$	Steady vector fields
$\mathbf{u}(\mathbf{x}, t)$	Time-dependent vector fields
$\bar{\mathbf{x}}$	Space-time coordinates $\bar{\mathbf{x}} = (\mathbf{x}, t)^\top$
$\bar{\mathbf{u}}(\bar{\mathbf{x}})$	Space-time vector fields $\bar{\mathbf{u}}(\bar{\mathbf{x}}) = (\mathbf{u}(\mathbf{x}, t), 1)^\top$
$\boldsymbol{\eta}, \mu$	Eigenvector with corresponding eigenvalue
$\boldsymbol{\xi}, \lambda$	Lyapunov vector with corresponding Lyapunov exponent
$a, b[, c]$	Variables a and b in the 2D case, additionally c in the 3D case

2 Fundamentals and Related Work

In this section, we introduce the flow visualization background for this thesis, and give an overview of related work. We first give a broad overview over visualization techniques for scalar and vector fields, and finally discuss concepts related to local features and vector field topology in detail, which are most closely related to our work.

2.1 Scalar Fields

A function $f : \Omega \to \mathbb{R}$ defined on an n-dimensional domain $\Omega \subset \mathbb{R}^n$ is called a scalar field. In the following, we assume f to be at least twice continuously differentiable, i.e., that f is $\mathcal{C}^2$. In scientific visualization, scalar fields represent continuous univariate data, which typically have either been measured or obtained from numerical simulation. Examples are pressure or speed (magnitude of velocity) in a fluid flow.

The direction of steepest ascend at each location $\mathbf{x} \in \Omega$ is given by the column vector of its partial derivatives,

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x}) \right)^\top, \tag{2.1}$$

which is called the gradient of f. The ∇-operator denotes the vector of partial derivative operators, $\nabla = (\partial/\partial x_1, \ldots, \partial/\partial x_n)^\top$.

The matrix of its second partial derivatives,

$$\mathbf{H}(\mathbf{x}) = \nabla\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1}(\mathbf{x}) & \ldots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) & \ldots & \frac{\partial^2 f}{\partial x_n \partial x_n}(\mathbf{x}) \end{pmatrix} \tag{2.2}$$

is called the Hessian of f. By the assumption that f is twice continuously differentiable, $\mathbf{H}$ is a real symmetric matrix and thus admits an orthonormal eigendecomposition. At a critical point $\mathbf{x}_c$, i.e., $\nabla f(\mathbf{x}_c) = \mathbf{0}$, these eigenvalues and eigenvectors correspond to

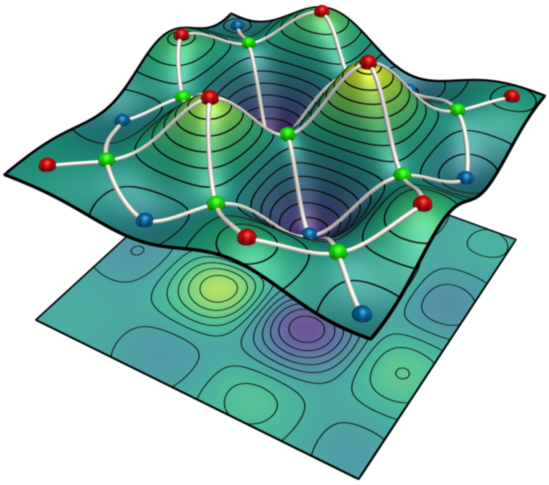


Figure 2.1: Visualization of a 2D scalar field $f : \mathbb{R}^2 \to \mathbb{R}$ as height map (top), i.e., the graph of f in 3D, together with its 2D contour plot (bottom, desaturated). Scalar values mapped to color (viridis color map, purple to yellow, from low to high values of f), shown together with isocontours (black lines). Morse–Smale complex shown on height map, consisting of critical points (maxima red, saddles green, minima blue), and separatrices (white lines), which are gradient lines of f that connect critical points.

the principal curvatures and principal directions of curvature of the graph of f [Cal10, Sec. 7.3], i.e., the n-dimensional manifold $\{(\mathbf{x}, f(\mathbf{x})) \mid \mathbf{x} \in \Omega\} \subset \mathbb{R}^{n+1}$.

2.1.1 Scalar Field Visualization

Direct visualization approaches map the scalars at each spatial location to a color or other property and render it directly. In the case of 2D scalar fields, or two-dimensional slices of 3D scalar fields, color maps are typically used to obtain a mapping between the scalar value and a color. A technique for direct visualization of 3D scalar fields is volume rendering [Eng+04; HHS93; HJ11]. This technique is based on accumulating color and opacity along view rays to obtain an image. Color and opacity are typically obtained from transfer functions, which, based on the specific dataset, need to be chosen suitably to reveal the structures of interest. Multi-dimensional transfer functions further take, e.g., the gradient of the scalar field taken into account [KKH02].

Indirect visualization methods are based on extracting features, resulting in geometric objects that represent the structure of the data. Isocontours [LC87] are a commonly used feature for the indirect visualization of scalar fields (e.g., black lines in Figure 2.1). The preimage $f^{-1}(\{c\})$ of a scalar value c, i.e., the set of all locations $\mathbf{x}$, such that $f(\mathbf{x}) = c$, is called an isocontour. By the implicit function theorem [BB56, Sec. 7.6], under the condition that f is $\mathcal{C}^1$ and $\nabla f(\mathbf{x}) \neq \mathbf{0}$, it is an ($n$-1)-dimensional manifold. For multivariate data, i.e., sets of scalar fields defined on the same domain, these have been generalized to fibers [Ble+20; Car+15; KTCG16] and feature level sets [JH18].

Beyond this, topology-based methods [Hei+16] are employed. A basic topological building block are critical points, which are isolated locations $\mathbf{x}_c$ where the gradient of the scalar field vanishes: $\nabla f(\mathbf{x}_c) = \mathbf{0}$. They can be classified using the eigenvalues of the Hessian matrix $\mathbf{H}(\mathbf{x}_c)$. If an eigenvalue is zero, the critical point is degenerate. Otherwise, if all eigenvalues are positive, it is a local minimum, if all eigenvalues are negative, it is a local maximum, and if there are eigenvalues of both signs, it is a saddle point. Data that, for example, exhibit numerical noise typically have a large number of critical points, and their topology thus needs to be simplified in order to produce insightful visualizations. Persistence, which measures the "duration" of pairs of critical points, is commonly employed for topological simplification [ELZ00], where critical point pairs are removed starting from the lowest persistence up to a user-defined threshold. We are going to employ such simplification to obtain a ground-truth from FTLE fields in Section 5.6.

Ridges can be considered a generalization of local extrema, in the sense that they represent extremal lines. There are several different co-existing definitions, each of which has advantages and disadvantages [LLSV99]. While for each definition counterexamples of ridges can be constructed that are not captured by the respective concepts, the most commonly used definitions are a local definition based on the Hessian [PS08] and a global definition based on watersheds [WG09]. The local definition due to Eberly et al. [Ebe+94], which is based on generalized extrema in directions of eigenvectors of the Hessian, will be discussed in detail in Section 2.5.2. Watersheds [Max70] are a global definition based on gradient lines, i.e., tangent curves of ∇f. They are defined as those gradient lines that separate the domain into basins of attraction. These are a subset of the separatrices in the Morse–Smale complex [DFIM15], which partitions the domain into cells of gradient curves connecting the same extrema. An example is shown in Figure 2.1. This partition can be obtained by extracting separatrices of the saddle-type critical points in the vector field topology of ∇f (see Section 2.6). A discussion about the difference between locally defined ridges and separatrices can be found, e.g., in the work of Sahner et al. [SWTH07, Sec. 3.2].

Providing even more reduced descriptions of scalar fields, topological descriptors [Yan+21] can be employed, which further lend themselves for comparative visualizations of sets of scalar fields.

In this thesis, we are going to encounter scalar fields as fields derived from vector fields, such as the velocity magnitude, or quantities derived from the flow map, as we discuss next. Since these fields are going to be higher-dimensional, we are mainly concerned with feature-based visualization, such as using ridge extraction.

2.2 Vector Fields

A map $\mathbf{u} : \Omega \times \mathcal{T} \to \mathbb{R}^n$ is called a time-dependent (or: unsteady, transient) vector field defined on the spatial domain $\Omega \subseteq \mathbb{R}^n$ and a temporal domain $\mathcal{T} \subseteq \mathbb{R}$. If $\mathbf{u}(\mathbf{x}, s) = \mathbf{u}(\mathbf{x}, t)$ for any $s, t \in \mathcal{T}$, the vector field is called steady (or: stationary), and we omit the time parameter: $\mathbf{u}(\mathbf{x})$. A time-dependent vector field defined on an interval $\mathcal{T} = [t_0, t_N]$ is called finite-time.

The dynamics associated with a vector field is the temporal evolution of initial values $(\mathbf{x}_0, t_0) \in \Omega \times \mathcal{T}$ described by the ordinary differential equations (ODEs)

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{u}(\mathbf{x}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \,. \tag{2.3}$$

The solution curve $\mathbf{x}(t)$ is also called a trajectory of $\mathbf{u}$. Such a system of ODEs is also called a dynamical system, which is said to have an underlying vector field. The study of dynamical systems [CK14; Wig90] is, however, mainly concerned with the evolution of initial conditions in the limit $t \to \infty$, where $\mathbf{u}$ is typically given as analytic function.

The flow of a vector field is the temporal evolution of initial values under Equation 2.3. It is described by the flow map $\boldsymbol{\phi}_{t_0}^{T}(\mathbf{x})$, which maps a location $\mathbf{x}$ and initial time t_0 to its position after advection time T, i.e., the location of the massless particle at time $t_0 + T$. For steady vector fields, we omit the parameter t_0 and write $\boldsymbol{\phi}^{T}(\mathbf{x})$.

Time-dependent vector fields can also be regarded as a steady vector field, where time is included as additional dimension in the state variable $\mathbf{x}$. Writing $\bar{\mathbf{x}} = (\mathbf{x}, t)^\top$, Equation 2.3 can be written as

$$\frac{\mathrm{d}\bar{\mathbf{x}}(s)}{\mathrm{d}s} = \frac{\mathrm{d}}{\mathrm{d}s}\begin{pmatrix}\mathbf{x}(s)\\ t(s)\end{pmatrix} = \begin{pmatrix}\mathbf{u}(\mathbf{x}(t), t)\\ 1\end{pmatrix}, \quad \bar{\mathbf{x}}(s_0) = \begin{pmatrix}\mathbf{x}(s_0)\\ t(s_0)\end{pmatrix} = \begin{pmatrix}\mathbf{x}_0\\ t_0\end{pmatrix}. \tag{2.4}$$

The resulting $(n{+}1)$-dimensional steady vector field $\bar{\mathbf{u}} = (\mathbf{u}, 1)^\top$is called the space-time vector field $\bar{\mathbf{u}}$, defined on the space-time domain $\Omega \times \mathcal{T}$.

The spatial derivative of a vector field is the $n \times n$ Jacobian matrix,

$$\nabla \mathbf{u} = (\nabla u_1, \dots, \nabla u_n)^\top, \tag{2.5}$$

which consists of the gradients of the vector components. Unlike the Hessian matrix of a scalar field, it generally is not symmetric and thus may exhibit complex eigenvalues. The Jacobian describes instantaneous behavior, i.e., the limit of advection time approaching zero $T \to 0$, of nearby trajectories in an infinitesimal neighborhood.

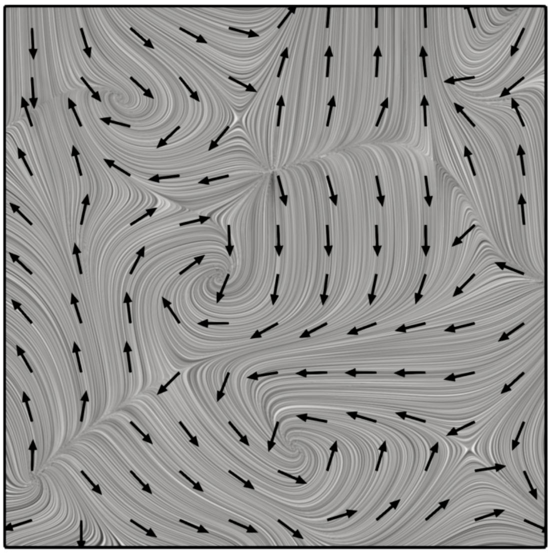


Figure 2.2: Steady 2D vector field $\mathbf{u}(\mathbf{x}) \in \mathbb{R}^2$. Arrow glyphs (black, fixed length) depict velocity direction at a set of locations, while line integral convolution (LIC, background) represents a dense set of tangent curves of $\mathbf{u}$.

2.2.1 Vector Field Visualization

By considering the velocity magnitude $\|\mathbf{u}\|$, direct visualization approaches for the resulting scalar field can be employed (Section 2.1.1). Considering the components of $\mathbf{u}$, or other derived quantities such as eigenvalues of $\nabla\mathbf{u}$, as a set of scalar fields, they can be mapped to glyphs [Bor+13], with the most prominent being the arrow glyph (see Figure 2.2), which encodes speed and direction.

Dense methods [Lar+04] aim at visualizing a dense set of trajectories (Equation 2.3). Line integral convolution (LIC) is a common method that involves smearing an initial random noise texture along trajectories ([CL93], Figure 2.2). This method has been further extended to unsteady [SK97] and 3D vector fields [RHTE99].

Integration-based geometric methods [McL+10] consider manifolds, such as surfaces or volumes that are advected with the flow. These also play an important role in illustrative flow visualization [Bra+12]. We are going to employ such stream and streak manifolds in the context of separatrices in vector field topology (Part II), and discuss them in detail in Section 2.4.

Finally, visualization based on feature extraction and feature tracking [Pos+03] is able to reduce the amount of data displayed. We discuss specific local features, which we are going to employ throughout this thesis, in detail in Section 2.5. Topology-based methods are commonly employed for both steady [LHZP07] and unsteady [Buj+20; Pob+11] vector fields. Steady vector field topology will be further discussed in Section 2.6, and unsteady vector field topology in Section 2.7.

2.3 Data Representation

Datasets representing continuous fields are typically not given by an analytic formula, but instead on discrete sample points, which are obtained by measurement or numerical simulation. The continuous domain of the data is partitioned into non-overlapping cells, where the sample points serve as nodes. Such a discretization into cells and their nodes is called a grid. They can be categorized into structured and unstructured grids. Examples are shown in Figure 2.3.

Structured grids have an implicit topology, where the cells can be indexed by tuples of integers $(i, j) \in \mathbb{Z}^2$ in 2D and $(i, j, k) \in \mathbb{Z}^3$ in 3D, such that cells with neighboring tuples in $\mathbb{Z}^n$ are adjacent in the grid. This implies that the cells are parallelotopes. For uniform grids, node positions can be stored implicitly by specifying the position of a single grid node, e.g., the origin, and the spacing $d_x, d_y[, d_z]$ in each direction. Examples for such grids are Cartesian grids, where d_x=d_y[=d_z]=1, regular grids, where $d_x, d_y[, d_z]$ are user-defined but constant within the grid, as well as rectilinear grids, where $d_x, d_y[, d_z]$ may vary depending on the index $(i, j[, k])$. These types of grids allow very fast lookup of a cell given a position in space. Curvilinear grids, on the other hand, explicitly store the position of each node, while still possessing implicit topology.

Unstructured grids explicitly store the connectivity between cells, and allow any types of cells. Common cell types are triangles or quads in 2D, and tetrahedra, pyramids, prisms or hexahedra in 3D. Since both connectivity and node positions have to be stored explicitly, these grids require more memory than their structured counterparts. Cell lookup is also more challenging, and requires specialized algorithms [GJ10; SML98].

In order to obtain a continuous scalar or vector field from a grid, the data stored at the grid nodes need to be interpolated. A common approach is to employ linear interpolation on the cell edges, and extend it to the entire cell. For triangle and tetrahedral cells, this leads to linear barycentric interpolation. In structured grids, a tensor-product approach is used, which leads to bilinear interpolation in 2D and trilinear interpolation in 3D. While interpolation within a cell only requires the data at its nodes, it comes at the cost of being only $\mathcal{C}^0$ on the cell edges, i.e., it is not continuously differentiable, since the interpolation functions on the interiors of two neighboring cells are independent, and thus have different derivatives near the cell edges. Higher-order interpolation schemes can overcome this limitation by taking a neighborhood of cells into account. However, this comes at increased computational costs, and typically also with a smoothing effect on the data, which may not be desired. Spline-based interpolation kernels are commonly employed [BG88; CR74; Kin+18; KTW07] in this context.

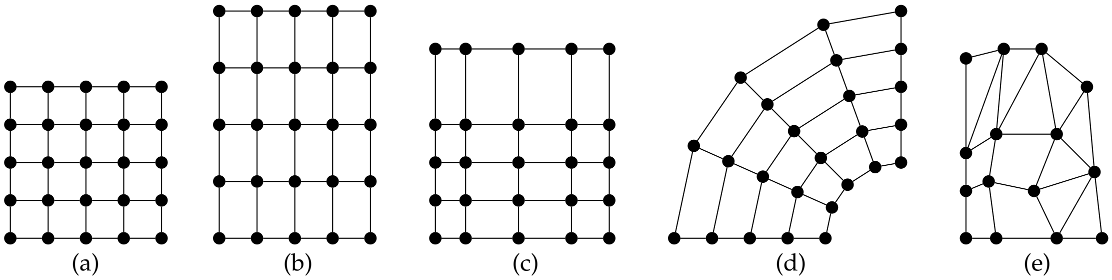


Figure 2.3: Different types of two-dimensional grids (black dots: grid nodes). Cartesian grids (a), regular grids (b), rectilinear grids (c), and curvilinear grids (d) are structured, i.e., have implicitly defined topology with regular connectivity. Unstructured grids (e) may consist of heterogeneous types of cells (here: triangles and quads), where topology needs to be defined explicitly.

Since they straightforwardly extend to higher dimensions (see Appendix A), and provide the most efficient data structure, we employ regular grids with multilinear interpolation throughout this thesis.

2.3.1 Derivatives on Structured Grids

Since we focus on $\mathcal{C}^0$ interpolation schemes, these do not lend themselves for obtaining derivatives of the data directly. Instead, we approximate derivatives on the grid nodes, and interpolate these. Due to its efficiency, this is a commonly used approach [STS09]. However, we note that derivatives obtained in this way are not consistent with the interpolation on the interior of the cells. This, again, can be overcome by employing higher-order interpolation schemes.

Finite differences is a commonly employed method for approximating derivatives on regular grids. For a one-dimensional scalar function f, where we denote by $\mathfrak{f}[i]$ the nodes of a regular grid with spacing d, the derivatives at the grid nodes are computed by forward, central, and backward differences as

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{\mathfrak{f}[i+1] - \mathfrak{f}[i]}{d}, \quad \frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{\mathfrak{f}[i+1] - \mathfrak{f}[i-1]}{2d}, \quad \frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{\mathfrak{f}[i] - \mathfrak{f}[i-1]}{d}. \tag{2.6}$$

In the case of vector- or matrix-valued functions, this approximation is carried out separately in each component of the function. For higher-dimensional grids, the derivatives in each spatial direction are obtained dimension-wise, i.e., by keeping the remaining coordinates fixed. While central differences are second-order approximations, forward and backward are of first order. The latter are often used on the grid boundaries, where only one neighboring data point is available.

Central differences can also be written as discrete convolution

$$(\mathfrak{f} * \mathfrak{g})[i] = \sum_{k=-N}^{N} \mathfrak{f}[i-k]\mathfrak{g}[k], \tag{2.7}$$

with N=1 and where $\mathfrak{g}$ takes the values $1/2d, 0, -1/2d$ at indices $-1, 0, 1$. This formulation allows for efficient computation of the derivatives on the entire grid, and extends to higher-order schemes by replacing the convolution kernel $\mathfrak{g}$ appropriately. Since for continuous convolution, defined as $(f * g)(x) = \int f(x-y)g(y)\mathrm{d}y$, the identity

$$\frac{\mathrm{d}}{\mathrm{d}x}(f * g) = f * \frac{\mathrm{d}g}{\mathrm{d}x} \tag{2.8}$$

holds [Bra86, Ch. 6], convolution with the analytically obtained derivative of a smoothing kernel results in accurate derivatives of the smoothened data. While we are mostly employing finite differences, convolution with derivatives of Gaussians are employed on a specific dataset in Section 5.6.3.

2.4 Integral Curves and Manifolds

Solutions $\mathbf{x}(t)$ of the system of ODEs (Equation 2.3) associated with a vector field $\mathbf{u}(\mathbf{x}, t)$ can be written as

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbf{u}(\mathbf{x}(s), s)\mathrm{d}s, \tag{2.9}$$

and are thus also called integral curves. Under the condition that $\mathbf{u}$ is uniformly Lipschitz continuous, these ODEs have unique solutions by the Picard–Lindelöf theorem [HNW93, Ch. I.7 and I.8].

There is a variety of numerical integration schemes for solving systems of ODEs that need to be chosen based on the properties of the ODE and the error requirements. In this thesis, we employ the the fourth-order Runge–Kutta scheme [HNW93, Ch. II.1], since it provides a good trade-off between accuracy and performance. Being an explicit scheme, it advances forward in time using a fixed step size Δt, and results in a discrete polyline $\mathbf{x}(t_0), \mathbf{x}(t_0 + \Delta t), \mathbf{x}(t_0 + 2\Delta t), \ldots$. More reliable results can be obtained using adaptive step sizes, where the integration error is monitored by additionally performing an integration step of higher order. Here, the embedded Runge–Kutta 4/5 scheme [DP80] is prominently used, which is able to partly reuse coefficients from the fourth-order step to obtain the fifth-order integration. This integration scheme can be used to further obtain a cubic spline representation, so-called dense output [Sha85], rather than a polyline of linear segments. This allows to obtain accurate results at specific points in time, which may not be reached exactly due to the adaptive step size. We are going to employ the Runge–Kutta 4/5 scheme with dense output for the computation of the fundamental solution matrix (Equation 2.26; see also Appendix B.2).

If an initial guess for $\mathbf{x}(t)$ is available, Equation 2.9 could also be solved by a fixed-point iteration, where the right-hand side of the equation is evaluated repeatedly until convergence. This method, however, has no advantage over Runge–Kutta methods in general, and is typically only employed to prove the Picard–Lindelöf theorem, where it is also called Picard iteration [HNW93, Ch. I.8]. A modification of the integral equation is, however, going to be solved using this method in Section 5.1, where we are going to be in a setting that provides suitable initial guesses.

As we are going to discuss next, different seeding strategies for initial particles for Equation 2.9 lead to different kinds of curves with different physical interpretations.

2.4.1 Streamlines

In steady vector fields, trajectories of the flow (Equation 2.3) are called streamlines. By the uniqueness theorem, every point $\mathbf{x}$ has exactly one streamline passing through it, i.e., no two streamlines can intersect. At locations $\mathbf{u}(\mathbf{x}_c) = \mathbf{0}$, multiple streamlines may converge toward the point $\mathbf{x}_c$, but the streamlines do not intersect geometrically.

Streamlines can also be obtained from time-dependent vector fields by freezing time at an instance $t = t_0$ (see Figure 2.4). In the space-time domain, they are tangent curves of the vector field $\bar{\mathbf{s}} = (\mathbf{u}, 0)^\top$. Here, the initial time t_0 in the extended space-time domain selects the time slice, which is regarded as a steady vector field. Since the last component of $\bar{\mathbf{s}}$ is zero, particles only advance in space but not in time.

2.4.2 Pathlines

Trajectories of time-dependent flow are called pathlines. As discussed in Section 2.2, they are streamlines of the space-time vector field $\bar{\mathbf{u}} = (\mathbf{u}, 1)^\top$. Since the last component of this vector field is one, particles advance in time at constant speed. Pathlines $\mathbf{x}(t)$ are thus a projection of the trajectory in the $(n+1)$-dimensional space-time domain onto the n-dimensional spatial domain. Thus, multiple pathlines may intersect in any point $\mathbf{x} \in \Omega$ (see Figure 2.4 for an example).

In steady vector fields, pathlines are equivalent to streamlines, since there is no dependence on time.

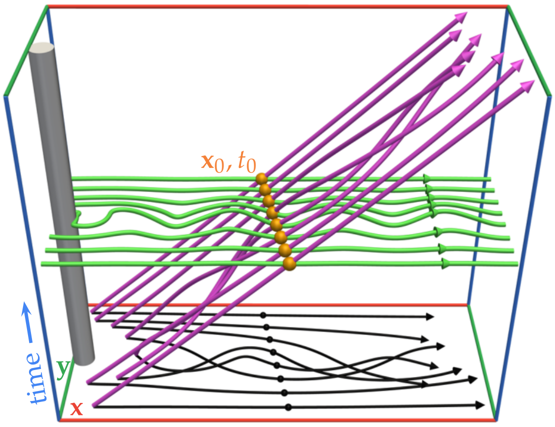


Figure 2.4: Space-time view of streamlines (green lines) and pathlines (magenta lines) in the 2D Cylinder Flow (Section 5.6.1), seeded (orange spheres) at a fixed point t_0 in time for varying spatial locations $\mathbf{x}_0$. Pathlines may intersect, when projected (black lines) onto 2D spatial domain.

2.4.3 Stream Surfaces and Stream Volumes

Streamlines can be extended to stream surfaces by replacing the initial value $\mathbf{x}_0$ with an initial curve $\mathbf{c}(s)$. Similarly, stream volumes can be obtained by seeding them from an initial surface $\boldsymbol{\mathcal{S}}(s, u)$. These concepts straightforwardly extend to path surfaces and path volumes, by replacing streamline with pathline integration. For brevity, we focus on stream manifolds in this section.

Stream manifolds are computed starting from a discretized initial curve or initial surface, e.g., an initial polyline or an initial triangle mesh. From the respective vertices, streamlines are computed by numerical integration. During integration, however, the quality of the mesh quickly degrades in typical flows. Edges may become too long, such that the continuous manifold is no longer well approximated by the linear segment, or too short, leading to duplicate vertices due to numerical precision. A typical approach is inserting new seeds by linear interpolation, once an edge becomes longer than a user-defined threshold, or removing seeds, once two incident edges become shorter than a threshold. While seeds inserted in this way do not lie on the analytical stream surface, this heuristic has been employed successfully, providing good performance. Hultquist [Hul90; Hul92] proposed an algorithm based on this method for triangle-based stream surfaces. This rather complicated algorithm, which is based on advancing a stream ribbon and requires complex data structures, can be simplified using a quad-based approach [MLZ09]. Stream volumes can be obtained in a similar fashion by integration of prism cells from an initial triangle mesh [MBC93], where triangles are subdivded or merged instead. We are going to employ these simple approaches throughout this thesis for the computation of invariant manifolds (see Section 2.6.3).

The stream surface algorithm by Hultquist was improved by Garth et al. [Gar+04; Gar+08], who employed different regularization schemes and based their method on arc-length parametrization using higher-order numerical integration. Further advanced techniques involve higher-order tesselation [SWS09], and incorporating vector field topology (Section 2.6) to overcome problems of these algorithms near critical points [PS09b; SRWS10]. Schulze et al. [SGRT12] proposed an algorithm that keeps the integration front orthogonal to the flow, thereby improving the mesh quality. A modification of the concept of as-perpendicular-as-possible surfaces [SRGT12] can be employed to grow flow tangential surfaces for obtaining stream surfaces without explicit integration of the flow [ESRT13]. Avoiding a triangulation altogether, Machado et al. [MSE14] proposed an image-based approach that computes a dense set of streamlines.

2.4.4 Streaklines

Continuously releasing particles over time at a fixed location $\mathbf{x}$ results in a streakline. Each particle moves along a different pathline, where the initial time continuously increases. In real-world experiments, streaklines can be observed by injecting smoke or dye into a fluid. Using the flow map, the streakline started from time t_0 at position $\mathbf{x}$, at current time t is the curve $\mathbf{c}^t(s) = \boldsymbol{\phi}_s^{t-s}(\mathbf{x})$, for $s \in [t_0, t]$.

Computing a streakline involves the advection of a polyline, where new seeding points at the point $\mathbf{x}$ are inserted at discrete time steps. Such a polyline undergoes the same stretching and contraction as the front of a stream surface, and needs to be refined in a similar fashion. Existing stream surface algorithms can be directly applied by computing a stream surface in the space-time flow $\bar{\mathbf{u}}$ seeded from the line between $(\mathbf{x}, t_0)^\top$ and $(\mathbf{x}, t_0 + T)^\top$. Time slices of such space-time stream surfaces then yield the streakline at each point in time, and the stream surface further lends itself to visualize the evolution of the streakline in space-time (see Figure 2.5). Streaklines can also be obtained as tangent curves of a derived (n+2)-dimensional vector field [WT10], which is based on evaluation of the flow map. Allowing the point $\mathbf{x}$ to move along a continuous curve leads to generalized streaklines [Wie+07], which we are going to employ mostly throughout this thesis.

2.4.5 Timelines

Injecting particles along a curve $\mathbf{c}_0(s)$ at a single point in time results in a timeline. The timeline seeded at time t_0 after advection time T is given by $\mathbf{c}^{t_0+T}(s) = \boldsymbol{\phi}_{t_0}^{T}(\mathbf{c}_0(s))$. Note that in fluid dynamics literature a time line is also called a material line.

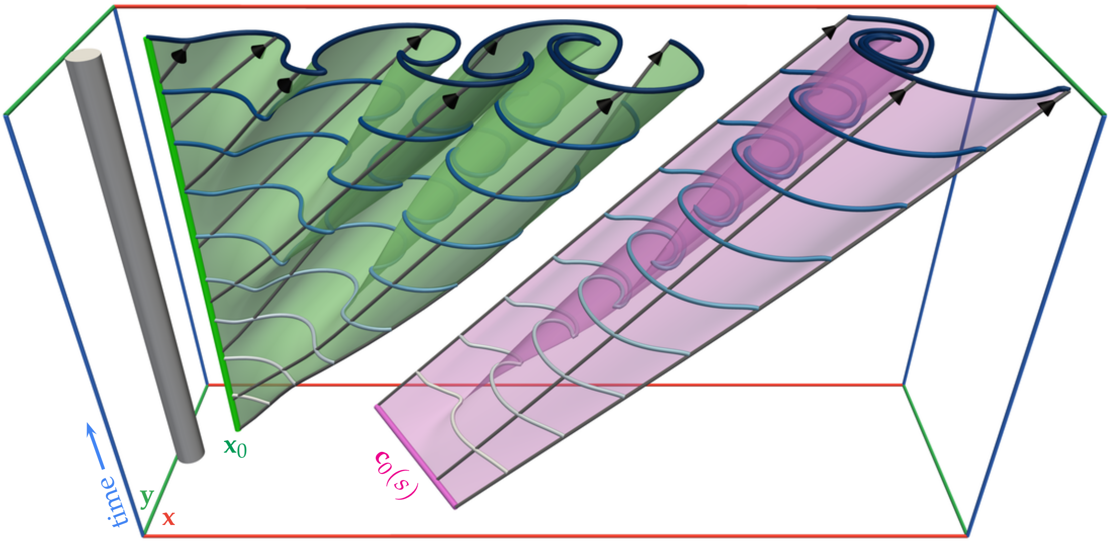


Figure 2.5: Streaklines and timelines in the 2D Cylinder Flow (Section 5.6.1) are obtained as time slices (lines, time in shades of blue) of space-time stream surfaces (green/pink). While streaklines are seeded from a time-parallel seeding line $\mathbf{x}_0$ (green line), timelines are seeded from a time-constant seeding line $\mathbf{c}_0(s)$ (pink line). The stream surfaces represent the temopral evolution of the time and streaklines. Pathlines (black) shown on the stream surfaces for reference.

Timelines can be computed as time slices of a space-time stream surface (Figure 2.5), similarly to streaklines, by seeding the stream surface at the curve $\bar{\mathbf{c}}_0(s) = (\mathbf{c}_0(s), t_0)^\top$. Again, Weinkauf et al. [WHT12] constructed a derived vector field, such that timelines are its tangent curves.

2.4.6 Time Surfaces and Streak Surfaces

Timelines and streaklines extend to surfaces by replacing the involved space-time seeding lines with space-time seeding surfaces. From these surfaces, space-time stream volumes are obtained, such that time slices represent the time and streak surfaces, depending on the shape of the seeding structure within the space-time domain (Figure 2.6). Again, in fluid dynamics literature, time surfaces are also called material surfaces.

In practice, however, computing such space-time stream volumes leads to prohibitively large meshes. Krishnan et al. [KGJ09] proposed an algorithm, where a triangle mesh is advected, and after each integration step a series of edge flips, collapses, and splits is performed to keep the mesh regular. The authors further propose a compressed representation that records the computed integral lines together with the sequence of edge operations. This information can be used to recover the entire evolution of the

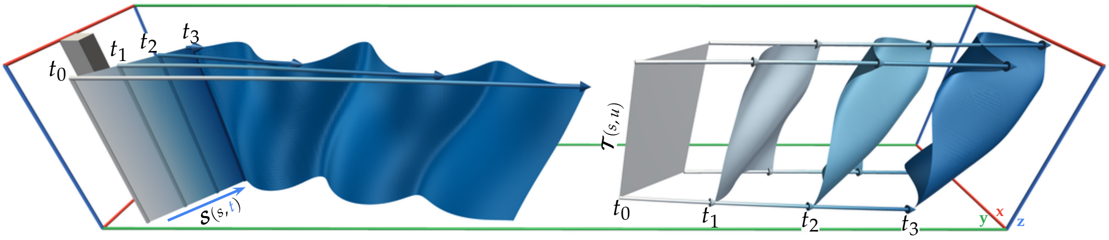


Figure 2.6: Streak surface (left, shown at time t_3) and time surfaces (right, shown at times t_0, t_1, t_3, t_3) in the time-dependent 3D Von Kármán Vortex Street (Section 5.6.3). Time is indicated in shades of blue (white to blue, from t_0 to t_3). While a time surface is seeded from a surface $\mathcal{T}(s, u)$ at a single instance of time t_0, a streak surface is obtained from a time-varying seeding surface $\mathcal{S}(s, t)$, which represents a line moving over time (tubes, shown at times t_0, t_1, t_3, t_3).

surface, and is thus equivalent to the computation of a space-time stream volume. We are going to employ this method to obtain time and streak surfaces in Chapter 5.

2.5 Local Features

We call features in scalar or vector fields local, if for each point $\mathbf{x} \in \Omega$ in the domain, the field and its derivatives at $\mathbf{x}$ are sufficient to decide if the point belongs to the feature. In practice, for a numerically stable and efficient computation, small neighborhoods $\mathcal{U}(\mathbf{x}) \subset \Omega$, e.g., a cell of the grid, are typically considered instead. In the case of time-dependent fields, either a temporal derivative or a small neighborhood in time may be further required. Local features have the advantage that their extraction can often be trivially parallelized, since only small pieces of the dataset are required at a time [AGL05; Ahr+01; Chi+10]. Examples for such features include critical points and isocontours in scalar fields, which, due to their local definition, can be visualized directly using volume rendering [Eng+04]. Kindelmann et al. [Kin+18] proposed a framework that allows for volume rendering and particle-based extraction of any local feature that can be formulated as generalized local extrema of a derived scalar field. In this thesis, however, we are interested in geometrically extracting features, and use these explicit representations for further computations. In the following, we give an overview of those local features that make up the basis of this thesis.

2.5.1 Parallel Vectors Operator

Line-type features [Rot00] in 3D scalar and vector fields can often be characterized by parallelism of two (derived) vector fields. This motivated Peikert and Roth [PR99] to formulate the parallel vectors (PV) operator, which for two vector fields $\mathbf{u}$, $\mathbf{w}$, defined on a common domain Ω, returns the set of locations $\mathbf{x}$ where $\mathbf{u}(\mathbf{x}) \parallel \mathbf{w}(\mathbf{x})$. Equivalently, this set can be defined as those locations where the cross product of the two vector fields is the zero vector, i.e.,

$$\{\mathbf{x} \in \Omega \mid \mathbf{u}(\mathbf{x}) \times \mathbf{w}(\mathbf{x}) = \mathbf{0}\}. \tag{2.10}$$

The authors [PR99] propose two algorithms for finding solution points on two-dimensional cell faces of a grid. On triangle faces with linear barycentric interpolation, Equation 2.10 is reduced to an eigenvector problem, and thus can be solved directly. For our generalization to arbitrary dimension and number of vector fields, we employ an algorithm based on Newton iterations within cell faces instead, which was proposed by Peikert and Roth for quad faces in 3D, because the eigenvector-based method only generalizes to arbitrary dimension in the case of two vector fields (case k=1 in our dependent vectors operator, Chapter 3).

A variety of features can be obtained in this way, among others, ridge lines and valley lines ([Ebe+94; PS08], Section 2.5.2), attachment lines and separation lines ([KHL99], Section 2.6.5), as well as vortex core lines and bifurcation lines ([DSL90; MSE13; Rot00; RP98; SH95], Section 2.5.5).

Works that go beyond direct application of the PV operator include the one by Ju et al. [JCWD14], who derive a parity test for the number of PV solutions. Fuchs et al. [Fuc+08] discuss an extension of PV features to unsteady flows, and Van Gelder and Pang [VP09] present an approach to finding PV lines based on root-finding, which can be applied to arbitrary dimensions, as well as cases where the dimension of the vectors does not match that of the domain. Pagot et al. [Pag+11] extend the PV operator to higher-order data. The case of vortices with vanishing longitudinal component is treated by Jung et al. [Jun+17], by transforming the PV problem into a ridge extraction problem. An extension of the PV operator to vector field ensembles is presented by Gerrits et al. [GRT18]. Oster et al. [ORT18a; ORT18b] extract PV lines in 3D second-order tensor fields, where not only the 3D location but also the eigenvector itself is unknown, leading to a 5D search space. Günther and Theisel [GT18a] compute PV lines in a 6D phase space of inertial dynamics, using a generalized cross product. Witschi and Günther [WG20] present an interactive approach for raycasting PV lines using the GPU.

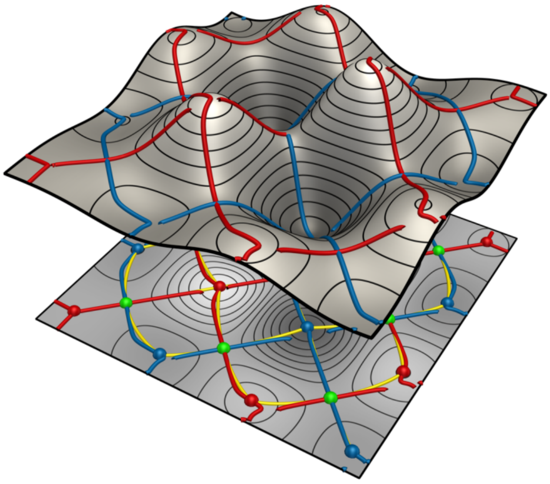


Figure 2.7: Ridges (red) and valleys (blue) in the 2D scalar field from Figure 2.1, extracted using the method proposed by Peikert and Sadlo [PS08], filtered by contour distance. Notice that, due to the local definition [Ebe+94], ridges and valleys do not necessarily connect saddles with extrema, and they generally do not follow gradient lines. Comparison with Morse–Smale complex (maxima red, saddles green, minima blue, separatrices yellow) shown at bottom.

2.5.2 Height Ridges

A local definition of height ridges is given by Eberly et al. [Ebe+94], based on the eigenvalues $\mu_1 \leq \mu_2[\leq \mu_3]$ and corresponding eigenvectors $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2[, \boldsymbol{\eta}_3]$ of the Hessian $\mathbf{H}(\mathbf{x})$ of a scalar field f. In the following, we give an overview of ridges and their extraction in 2D and 3D scalar fields (see Figure 2.7 for an example). Valleys are defined as ridges in $-f$, i.e., by replacing f with $-f$ in the definition of ridges. A generic definition for arbitrary dimensions will be discussed in Section 3.1.2.

A codimension-one ridge, i.e., ridge line in 2D and ridge surface in 3D, is defined as those locations $\mathbf{x}$ where the gradient of f is perpendicular to the minor eigenvector, which needs to have negative eigenvalue, i.e., $\nabla f(\mathbf{x}) \cdot \boldsymbol{\eta}_1(\mathbf{x}) = 0$ and $\mu_1 < 0$. The former equation can be regarded as an isocontour of a derived scalar field.

In 3D scalar fields, additionally ridges lines of codimension two exist. They are locations where, in addition to the conditions for a ridge surface, conditions on the medium eigenvector and eigenvalue, $\nabla f(\mathbf{x}) \cdot \boldsymbol{\eta}_2(\mathbf{x}) = 0$ and $\mu_2 < 0$, hold. Since the eigenvectors form a orthogonal basis, the two orthogonality conditions on ∇f can be simplified to the parallel vectors condition $\nabla f(\mathbf{x}) \parallel \boldsymbol{\eta}_3(\mathbf{x})$.

All of the above can be extracted by using a modification of the marching cubes [LC87] algorithm, or the parallel vectors operator, respectively, where the conditions on the eigenvalues are incorporated by filtering the resulting geometry. Since eigenvectors are only defined up to scalar multiple, they need to be oriented consistently within each cell or cell face considered by the extraction algorithms. Morse [Mor95] determines correspondences between sets of eigenvector orientations within each cell by comparing angles. Furst and Pizer [FP01] propose to use principal component analysis (PCA) to obtain consistent orientations, which we employ in our generic extraction algorithm in Chapter 3. More advanced techniques include the one by Kindlmann et al. [KTW06], who track eigenvectors along cell edges, which are subsampled to account for the non-linearity of the eigenvectors of the linearly interpolated Hessian matrix. Peikert and Sadlo [PS08] avoid computation of eigenvectors by extracting a superset of the ridges using marching cubes, which, however, suffers from false negatives due to the high non-linearity of the derived scalar field. By incorporating degenerate lines [ZP04] in the Hessian tensor field, which are part of the boundaries of ridge surfaces [Dam98], Schultz et al. [STS09] propose a more exact algorithm for the extraction of ridge surfaces, which further avoids orientation of eigenvectors.

Note that all of the issues in these ridge extraction approaches are caused by using a grid for efficient extraction. E.g., the sampling-based approach by Barakat et al. [BAT11] and the raycasting approach by Kindlmann et al. [Kin+18] do not suffer from these problems, since the derivatives and eigenvectors are evaluated pointwise.

2.5.3 Feature Flow Fields

Theisel and Seidel [TS03] proposed feature flow fields as a framework for reformulating features in vector fields as tangent curves of a derived vector field. Its applications include continuous tracking of features over time, as well as analysis of their bifurcations.

The feature flow field of a time-dependent vector field $\mathbf{u}(\mathbf{x}, t)$ is a derived vector field, such that $\mathbf{u}(\mathbf{x}, t)$ is constant along its tangent curves. Its main purpose is tracking and bifurcation analysis of critical points, since, by construction, $\mathbf{u}(\mathbf{x}, t) = \mathbf{0}$ along tangent curves of the feature flow field seeded at critical points $\mathbf{u}(\mathbf{x}_0, t_0) = \mathbf{0}$. For its construction, the authors consider the components of the vector field $\mathbf{u} = (u_1, u_2[, u_3])^\top$. The $(n{+}1)$-dimensional feature flow field $\mathbf{f}$ is constructed to be perpendicular to the space-time gradients $\nabla \bar{u}_i = (\nabla u_i, \partial u_i / \partial t)^\top$. In the 2D case, this can be achieved using the cross product, i.e., $\mathbf{f} = (\nabla \bar{u}_1) \times (\nabla \bar{u}_2)$. For the 3D case, a four-dimensional vector field $\mathbf{f}$ can similarly be derived. Note that the formula provided by the authors [TS03] can be reformulated using the wedge product, as defined in Section 3.1.1, which results in the equivalent expression $\mathbf{f} = (\nabla \bar{u}_1) \wedge (\nabla \bar{u}_2) \wedge (\nabla \bar{u}_3)$.

Another perspective is obtained by considering the space-time Jacobian of $\mathbf{u}$, which is the block matrix

$$\nabla \bar{\mathbf{u}} = \begin{pmatrix} \nabla \mathbf{u} & \mathbf{u}_t \\ \mathbf{0} & 0 \end{pmatrix}. \tag{2.11}$$

Assuming that $\nabla \mathbf{u}$ has full rank the nullspace, or eigenspace associated with eigenvalue zero, of $\nabla \bar{\mathbf{u}}$ is spanned by $\mathbf{f} = (-\nabla \mathbf{u}^{-1} \mathbf{u}_t, 1)^\top$, since the remaining eigenvectors are $(\boldsymbol{\eta}_i, 0)^\top$, where $\boldsymbol{\eta}_i$ are the eigenvectors of $\nabla \mathbf{u}$. This can be easily verified by straightforward calculation [GST16]. By construction, this nullspace precisely describes those directions, in which $\mathbf{u}$ does not vary, i.e., the directions of the feature flow field. As we are going to discuss in Sections 2.5.4 and 2.5.5, the feature flow field describes a Galilean-invariant frame of reference that is naturally obtained by applying the parallel vectors operator to the space-time vector field.

While Theisel and Seidel [TS03] further formulate a feature flow field for parallel vectors solutions, throughout this thesis, by the feature flow field of a vector field, we mean the one for tracking critical points as outlined above. Theisel et al. [The+05] provide an improved formulation of the parallel vectors feature flow field, where a pair of feature flow fields is employed to obtain parallel vectors surfaces, which describe the temporal evolution of PV lines over time. In a similar fashion, Weinkauf et al. [WTHS06] analyze bifurcations of critical points in 2D vector fields that dependent on two parameters instead of one. By incorporating higher-order derivatives, Weinkauf et al. [WTVP11] propose stable feature flow fields, in which the sought features are attractors, i.e., numerical integration is stable. Reinighaus et al. [RKWH11] avoid numerical integration altogether by using a discrete formulation.

2.5.4 Frames of Reference

In time-dependent vector fields that describe a fluid flow, an important property of a feature is its behavior under changes of frames of reference. A feature extraction technique, which is physically meaningful, must come to the same conclusion no matter in which frame of reference the dataset is observed. Since a feature moving through space can equivalently be regarded as a stationary feature with moving observer, such a reference frame invariance is essential for the analysis of unsteady flow.

By a change in frame of reference, we mean that the spatial coordinates $\mathbf{x}$ are transformed to $\mathbf{x}^*$ via

$$\mathbf{x}^* = \mathbf{R}(t)\mathbf{x} + \mathbf{b}(t), \tag{2.12}$$

where $\mathbf{R}(t)$ is a time-dependent rotation matrix, and $\mathbf{b}(t)$ a time-dependent translation. Invariance under such a coordinate change [TN04, Sec. 17], which is called objective, means for scalar fields $f(\mathbf{x}, t)$, vector fields $\mathbf{u}(\mathbf{x}, t)$, and tensor field $\mathbf{T}(\mathbf{x}, t)$ that

$$f^*(\mathbf{x}^*, t) = f(\mathbf{x}, t), \tag{2.13}$$

$$\mathbf{u}^*(\mathbf{x}^*, t) = \mathbf{R}(t)\mathbf{u}(\mathbf{x}, t), \tag{2.14}$$

$$\mathbf{T}^*(\mathbf{x}^*, t) = \mathbf{R}(t)\mathbf{T}(\mathbf{x}, t)\mathbf{R}(t)^\top. \tag{2.15}$$

Reference frame invariance under all possible objective transforms is called *objectivity*. An example for an objective quantity is the strain tensor $\mathbf{S} = (\nabla\mathbf{u} + \nabla\mathbf{u}^\top)/2$, and its eigenvalues and eigenvectors, as well as its derivatives.

A weaker form of observer invariance is *Galilean invariance*. It is obtained by restricting Equation 2.12 to the special case $\mathbf{R}(t) = \mathbb{1}$, and $\mathbf{b}(t) = \mathbf{c}_0 + t\mathbf{c}$, i.e.,

$$\mathbf{x}^* = \mathbf{x} + \mathbf{c}_0 + t\mathbf{c}, \tag{2.16}$$

which are translations at constant speed. Galilean-invariant quantities include the Jacobian $\nabla\mathbf{u}$ and acceleration $\mathbf{a} = (\nabla\mathbf{u})\mathbf{u} + \mathbf{u}_t$.

See Figure 2.8 for examples of different frames of reference.

Optimal Frames of Reference

Instead of explicitly constructing features with Galilean invariance or objectivity in mind, a frame of reference can be computed that itself is invariant under changes of observers. Any feature extracted from the resulting transformed vector field inherits the observer invariance of such a computation scheme. Since, however, features typically move in different directions and at different speeds, no single observer would be able to capture them at once [Lug79], but instead a continuous field of observers needs to be considered. Günther et al. [GGT17] propose to locally determine objective frames of reference by minimizing the observed temporal derivative of the vector field, i.e., by solving at each point $\mathbf{x} \in \Omega$ of the domain,

$$\int_{\mathcal{U}(\mathbf{x})} \|\mathbf{u}_t^*(\mathbf{x}^*, t)\|^2 dV \to \min \tag{2.17}$$

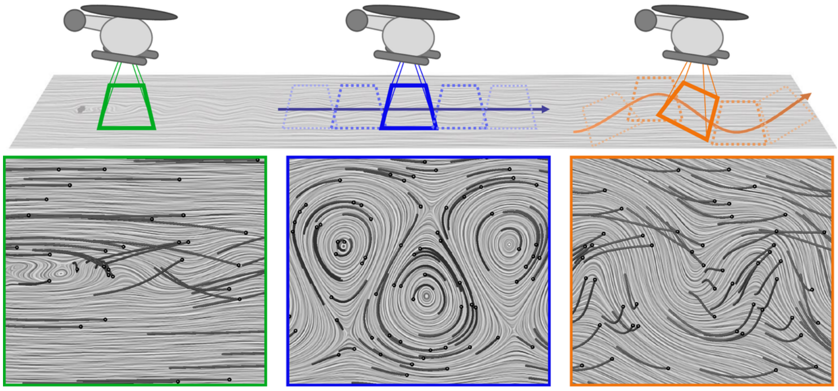


Figure 2.8: Observers (top) moving along different paths in the 2D Cylinder Flow (Section 5.6.1). Green: no movement ("lab frame"), blue: linear path (Galilean), orange: curved path (objective). The observed flow is not invariant (bottom: LIC of a time slice, observed pathlines in black), and thus shows different structures in each frame of reference. Image © 2017 ACM, reproduced, with permission, from [GGT17]; permission conveyed through Copyright Clearance Center, Inc.

for the parameters $\mathbf{R}, \mathbf{b}$ in Equation 2.12, using a neighborhood $\mathcal{U}(\mathbf{x})$ as regularizer. Since the resulting transformed vector field $\mathbf{w} = \mathbf{u}^* \mid_{t=t_0}$ is related to $\mathbf{u}$ by

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{w}(\mathbf{x}, t) + \mathbf{f}(\mathbf{x}, t), \tag{2.18}$$

this optimal frame of reference can also be interpreted [BG20] as a decomposition of $\mathbf{u}$ into a steady frame $\mathbf{w}$ and ambient motion $\mathbf{f}$. This is motivated by the fact that for Galilean transforms Equation 2.17 has the feature flow field $\mathbf{f} = -\nabla\mathbf{u}^{-1}\mathbf{u}_t$ as analytic solution [GT20]. Günther and Theisel [GT20] extended their method by allowing for affine transformations and similarity transformations, where the resulting frames of references are called affine invariant and similarity invariant, respectively. Baeza Rojo and Günther [BG20] further allowed any displacement by a diffeomorphism, i.e., an invertible differentiable function with differentiable inverse, $\mathbf{x}^* = \mathbf{x} + \mathbf{F}(\mathbf{x}, t)$, where we are going to call the resulting frame of reference *displacement invariant*. Note that only in the case of affine transformations, Günther and Theisel [GT20] provided analogous definitions to Equations 2.13 to 2.15. We are going to employ these concepts for the local candidate extraction in our time-dependent vector field topology (Chapter 5).

A similar, but global method was proposed by Hadwiger et al. [HMTR18], who solved for the transformed vector field $\mathbf{u}^*$ directly over the entire space-time domain. In a follow-up work, Rautek et al. [Rau+20] extended the method to vector fields defined on arbitrary two-dimensional manifolds, where they generalize the definition of objectivity on Euclidean spaces to arbitrary Riemannian manifolds. Recently, Haller [Hal21]

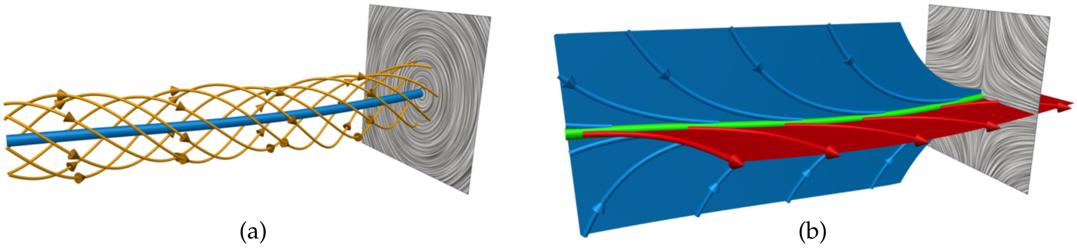


Figure 2.9: By the reduced velocity criterion [SH95], the feature line exhibits a critical point in the projected flow (LIC) orthogonal to an eigenvector of $\nabla \mathbf{u}$. A vortex core line ((a), blue) is obtained by requiring complex eigenvalues, which correspond to a plane of rotation (orange streamlines). Similarly, a bifurcation line ((b), green, after refinement [MSE13]) is obtained by real eigenvalues with alternating signs, which correspond to stable (blue) and unstable (red) stream surfaces. Note that feature lines with non-zero curvature are not necessarily streamlines.

pointed out conceptual limitations of approaches based on the minimization problem (Equation 2.17), which has since been disputed by the original authors [The+21].

2.5.5 Vortex Core Lines and Bifurcation Lines

There is a variety of different vortex definitions and extraction techniques, which in a recent survey Günther and Theisel [GT18b] classify into region-based, line-based, integration-based methods, and boundary extraction methods. In this thesis, we focus on the notion of a vortex core line based on the reduced velocity criterion due to Sujudi and Haimes [SH95]. See Figure 2.9 for an example.

The authors define a vortex core line in a steady 3D vector field as those locations where the projection of the velocity onto the plane of rotation is zero, i.e., the flow is entirely governed by non-rotating motion. In this definition, the plane of rotation is spanned by a pair of complex conjugate eigenvectors of the Jacobian $\nabla \mathbf{u}$, and regions where the Jacobian exhibits no complex eigenvalues are omitted. Denoting the only real eigenvector by $\boldsymbol{\eta}_r$, the condition of the reduced velocity criterion is

$$\mathbf{u}(\mathbf{x}) - (\mathbf{u}(\mathbf{x}) \cdot \boldsymbol{\eta}_r(\mathbf{x}))\boldsymbol{\eta}_r(\mathbf{x}) = \mathbf{0}. \tag{2.19}$$

Using the parallel vectors operator, this condition can be reformulated as

$$\mathbf{u}(\mathbf{x}) \parallel \boldsymbol{\eta}_r(\mathbf{x}) \quad \Rightarrow \quad \mathbf{u}(\mathbf{x}) \parallel \nabla \mathbf{u}(\mathbf{x})\mathbf{u}(\mathbf{x}), \tag{2.20}$$

where furthermore equivalence holds in regions with complex eigenvalues of $\nabla \mathbf{u}$, and which we are going base our generalize to arbitrary dimensions on in Section 3.1.3. For

2D steady vector fields, this condition becomes $\mathbf{u} = 0$, i.e., it reduces to spiral-type 2D critical points (see Section 2.6.1).

Being defined using the velocity $\mathbf{u}$, this criterion is not Galilean-invariant. By replacing the role of streamlines with pathlines in this concept, Weinkauf et al. [WSTH07] derive Galilean-invariant vortex core lines. Since in the space-time vector field $\bar{\mathbf{u}} = (\mathbf{u}, 1)^\top$, streamlines represent pathlines of the time-dependent flow, the authors propose to apply the Sujudi–Haimes criterion to $\bar{\mathbf{u}}$ in the 2D case. Recall (Section 2.5.3) that the only eigenvector of the space-time Jacobian $\nabla\bar{\mathbf{u}}$ with last component non-zero is the vector $(\mathbf{f}, 1)^\top$, where $\mathbf{f}$ is the feature flow field of $\mathbf{u}$. Therefore, we obtain

$$\bar{\mathbf{u}} \parallel (\nabla\bar{\mathbf{u}})\bar{\mathbf{u}} \quad \Leftrightarrow \quad \exists\alpha : \begin{pmatrix}\mathbf{u}\\1\end{pmatrix} = \alpha \cdot \begin{pmatrix}\mathbf{f}\\1\end{pmatrix} \quad \Leftrightarrow \quad \mathbf{u} - \mathbf{f} = \mathbf{0}, \tag{2.21}$$

i.e., a critical point in the Galilean-invariant frame of reference defined by the feature flow field. In the 3D case, Weinkauf et al. [WSTH07] require that $\bar{\mathbf{u}}$ lies in the plane spanned be the two real eigenvectors of $\bar{\mathbf{u}}$. By a similar computation, this coplanarity condition reduces to the parallel vectors condition

$$(\mathbf{u} - \mathbf{f}) \parallel \nabla\mathbf{u}(\mathbf{u} - \mathbf{f}), \tag{2.22}$$

i.e., the Sujudi–Haimes criterion in the the Galilean-invariant frame of reference defined by the feature flow field. Note that the Jacobian $\nabla\mathbf{u}$ is unchanged by a Galilean observer (Equation 2.15). A general coplanar vectors operator is going to be the special case $n = 4, k = 2$ in our dependent vectors operator (Chapter 3).

Bifurcation lines, as proposed by Perry and Chong [PC87], can be seen as a "saddle-type counterpart" to vortex core lines, which exhibit two-dimensional attracting and repelling manifolds. They are obtained by Machado et al. [MSE13] using a modification of the Sujudi–Haimes criterion [Rot00]. Instead of requiring the presence of complex eigenvalues, their presence is excluded, and the eigenvector belonging to the medium eigenvalue is used instead, i.e., $\mathbf{u} \parallel \boldsymbol{\eta}_2$, where the real eigenvectors $\boldsymbol{\eta}_i$ of $\nabla\mathbf{u}$ are ordered by their real eigenvalues, $\mu_1 \leq \mu_2 \leq \mu_3$. The condition that a bifurcation line has stream surfaces, which converge toward it in either forward- or backward-time, implies that it is a streamline segment. Therefore, Machado et al. [MSE13] propose a local refinement scheme that corrects the parallel vectors line toward a streamline. Their algorithm iteratively performs gradient descend steps at each vertex of the candidate line. Within the plane orthogonal to its tangent, each vertex is corrected toward the location that minimizes the angle between the tangent and the vector field $\mathbf{u}$. To make this local

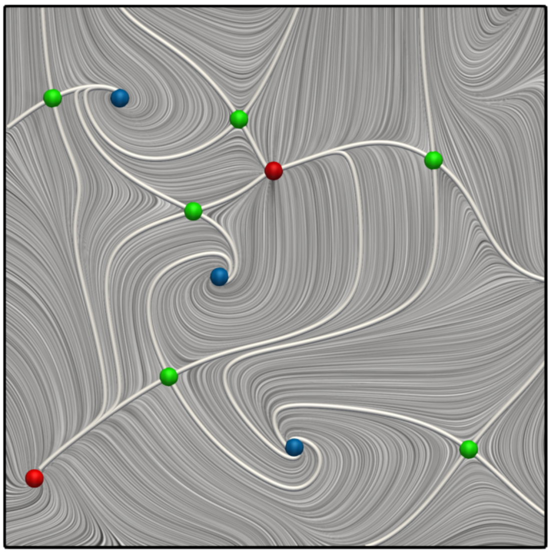


Figure 2.10: 2D vector field topology defined by critical points (red: sources, green: saddles, blue: sinks) and separatrices (white lines) of the vector field from Figure 2.2. LIC (background) shows the separating property of the separatrices: no streamline is able to cross a separatrix, and the flow within each region exhibits similar behavior.

refinement numerically feasible, the step size of the gradient descend is continuously reduced, and the planes are shifted to the separating planes of the neighboring vertices.

Similarly to vortex core lines, in time-dependent vector fields Galilean-invariant bifurcation lines can be extracted by applying the concept to the space-time vector field [MBES16]. The resulting pathlines represent hyperbolic trajectories, and will be discussed in detail in Section 2.7.3. In Chapter 5, we are going to compare our method for refinement of 2D hyperbolic trajectories to the method proposed by Machado et al. [MSE13], as outlined above. Furthermore, in the 3D case, we employ a concept similar swirling particle cores [WSTH07] to obtain candidates for hyperbolic path surfaces.

2.6 Vector Field Topology

A widely employed feature for the visualization of 2D and 3D steady vector fields is vector field topology [HH89; HH91]. The resulting topological skeleton provides a reduced representation of the qualitative properties of the flow, by separating the domain into regions of similar flow behavior (Figure 2.10).

This concept is based on the notion of topological equivalence of dynamical systems [CK14, Ch. 2.1]. Two steady vector fields $\mathbf{u}(\mathbf{x})$, $\mathbf{v}(\mathbf{x})$ with corresponding flow maps $\boldsymbol{\phi}^T(\mathbf{x})$, $\boldsymbol{\psi}^T(\mathbf{x})$ are called *topologically equivalent*, if there exists a homeomorphism $\mathbf{h} : \Omega \to \Omega$, i.e., a continuous map that possesses a continuous inverse, such that $\mathbf{h}(\boldsymbol{\phi}^T(\mathbf{x})) = \boldsymbol{\psi}^T(\mathbf{h}(\mathbf{x}))$ for all $\mathbf{x} \in \Omega$, $T \in \mathbb{R}$. This means that trajectories are mapped to trajectories, and that the direction of time is preserved. The following commutative diagram illustrates this:

$$\begin{tikzcd}
\Omega \arrow[r, "\boldsymbol{\phi}^T"] \arrow[d, "\mathbf{h}"'] & \Omega \arrow[d, "\mathbf{h}"] \\
\Omega \arrow[r, "\boldsymbol{\psi}^T"] & \Omega
\end{tikzcd}$$

The main focus of topology are properties that are unchanged under such homeomorphisms. For numerical computation, or in the context of numerical datasets, we further require that the structures cannot be destroyed by small numerical perturbations, i.e., we require them to be structurally stable. Specifically, $\mathbf{u}$ is *structurally stable*, if for any numerical perturbation $\mathbf{v}$ of $\mathbf{u}$, there exists a homeomorphism $\mathbf{h} : \Omega \to \Omega$, which maps trajectories of $\mathbf{u}$ to trajectories of $\mathbf{v}$. That is, given $\mathbf{v}$ with $\|\mathbf{u} - \mathbf{v}\| < \epsilon$ for a suitable norm and ϵ sufficiently small, there exists $\mathbf{h}$, such that $\mathbf{h}(\boldsymbol{\phi}^T(\mathbf{x})) = \boldsymbol{\psi}^T(\mathbf{x})$ for all $\mathbf{x}, T$.

In the remainder of this section, we discuss structurally stable topological properties, which together with their invariant manifolds make up the topological skeleton.

2.6.1 Critical Points

A critical point $\mathbf{x}_c$ is an isolated zero of the vector field, i.e., $\mathbf{u}(\mathbf{x}_c) = \mathbf{0}$ and $\det \nabla\mathbf{u}(\mathbf{x}_c) \neq 0$. Critical points are classified in terms of the linearization of the vector field in their neighborhood, i.e., by means of the eigenvalues of $\nabla\mathbf{u}(\mathbf{x}_c)$. This corresponds to a first-order approximation using a Taylor series expansion around the critical point,

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}(\mathbf{x}_c) + \nabla\mathbf{u}(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) + \mathcal{O}(\|\mathbf{x} - \mathbf{x}_c\|^2), \tag{2.23}$$

$$\mathbf{u}(\mathbf{x}) \approx \nabla\mathbf{u}(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c). \tag{2.24}$$

Classification in 2D

For $n = 2$, $\nabla\mathbf{u}(\mathbf{x}_c)$ is a real 2×2 matrix, exhibiting either two real or a pair of complex conjugate eigenvalues. In both cases, the real part indicates the inflow/outflow behavior of the vector field along the respective eigenvector, and a nonzero imaginary part indicates rotational behavior. Table 2.1 summarizes the five types of critical points in 2D vector fields (which can be reduced to the three types repelling node, repelling focus, and saddle, if flow reversal is allowed). Notice that complex eigenvectors represent eigenplanes (Tables 2.1d and 2.1e), which span $\mathbb{R}^2$; moreover, only one of these configurations separates different regions, i.e., that of type saddle (Table 2.1c). We indicate non-separating eigenvectors in these illustrations by dashed representation (leading to

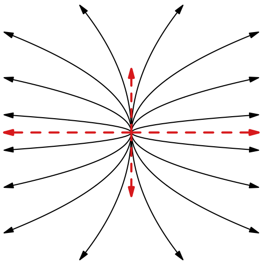
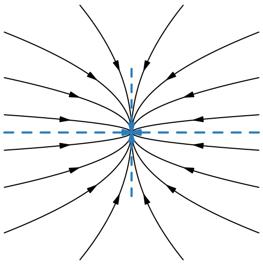
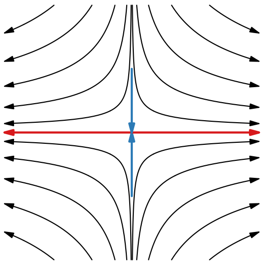
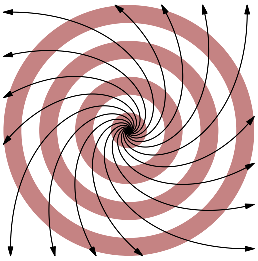
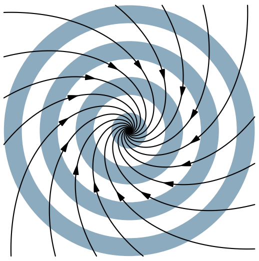
	$\rho_{1,2} > 0$	$\rho_{1,2} < 0$	$\rho_1\rho_2 < 0$
$\gamma = 0$	 (a) repelling node	 (b) attracting node	 (c) saddle
$\gamma = 1$	 (d) repelling focus	 (e) attracting focus	—

Table 2.1: Types of critical points in 2D vector fields, classified by the eigenvalues of $\nabla \mathbf{u}$, with real parts ρ_1, ρ_2, number of complex pairs γ, and ρ-scaled eigenvectors (unstable in red, stable in blue, non-separating are dashed).

circular bands for eigenplanes), whereas separating eigenvectors (and the manifolds they might span) are depicted with solid representation.

Classification in 3D

For 3D vector fields, $\nabla \mathbf{u}(\mathbf{x}_c)$ is a real 3×3 matrix, exhibiting either three real, or one real and a pair of complex conjugate eigenvalues. Again, the real part indicates the inflow/outflow behavior of the vector field along the respective eigenvector, and a nonzero imaginary part indicates rotational behavior. Table 2.2 summarizes the eight resulting types (which can be reduced to the four types source, saddle, spiral source, and spiral saddle, if flow reversal is allowed). Again, we have cases with eigenplanes (Tables 2.2d to 2.2f) indicating rotation. In 3D, we have two separating configurations, the saddle and spiral saddle (Tables 2.2c and 2.2f). These saddle-type cases can be further classified with respect to the number of incoming/outgoing directions. Table 2.2c depicts the case with one negative eigenvalue and two positive ones ($\rho_1\rho_2 < 0$), also denoted 1:2 saddle (1-manifold in, 2-manifold out). If we revert the direction of the vector field, we obtain the 2:1 saddle configuration ($\rho_2\rho_3 < 0$), exhibiting in our scheme a

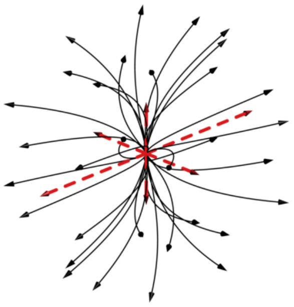
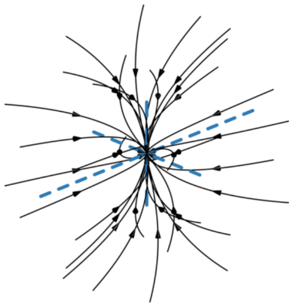
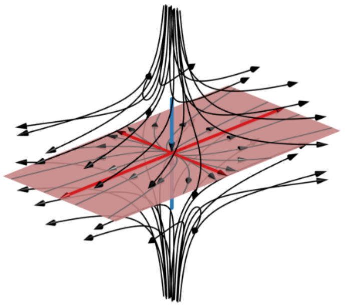
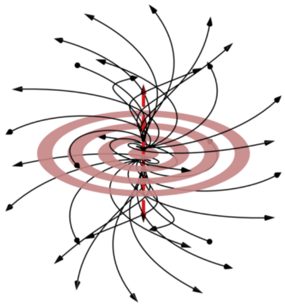
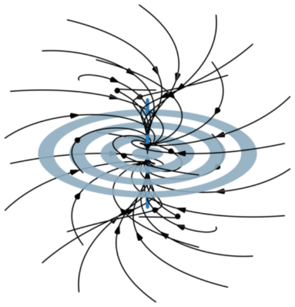
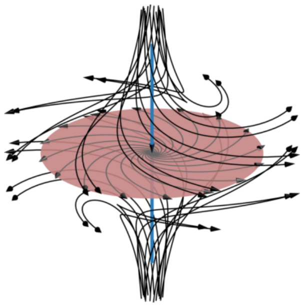
	$\rho_{1,2,3} > 0$	$\rho_{1,2,3} < 0$	$\rho_1\rho_3 < 0$
$\gamma = 0$	 (a) source	 (b) sink	 (c) 1:2 saddle
$\gamma = 1$	 (d) spiral source	 (e) spiral sink	 (f) 1:2 spiral saddle

Table 2.2: Types of critical points in 3D vector fields, based on the eigenvalues of $\nabla\mathbf{u}$, with real parts $\rho_{1,2,3}$ (ascending sorting), complex pairs γ, and ρ-scaled eigenvectors (repelling red, attracting blue, non-separating dashed). Notice that for the saddles, only $\rho_1\rho_2 < 0$ is illustrated.

blue 2-manifold and a red 1-manifold. Accordingly, there are 1:2 spiral saddles, as well as 2:1 spiral saddles, which we propose to abbreviate as 1:s2 and s2:1 saddles, the "s" indicating rotation of the respective manifold (this will become useful for the 4D cases).

Interestingly, we observe the cases from 2D fields in the cases in 3D fields. For example, the vector field on the separating red surface in Table 2.2c (see gray lines) exhibits a 2D repelling node (Table 2.1a), and the red surface in Table 2.2f exhibits a 2D repelling focus (Table 2.1d). Many other sections through the 3D cases exhibit respective 2D cases, in particular in sections spanned by the 3D eigenvectors. For example, the section spanned by the attracting (blue) eigenvector and one of the repelling (red) eigenvectors in Table 2.2c exhibits a 2D saddle (Table 2.1c). We will see that this principle continues through the analysis of critical points in 4D (Section 4.1).

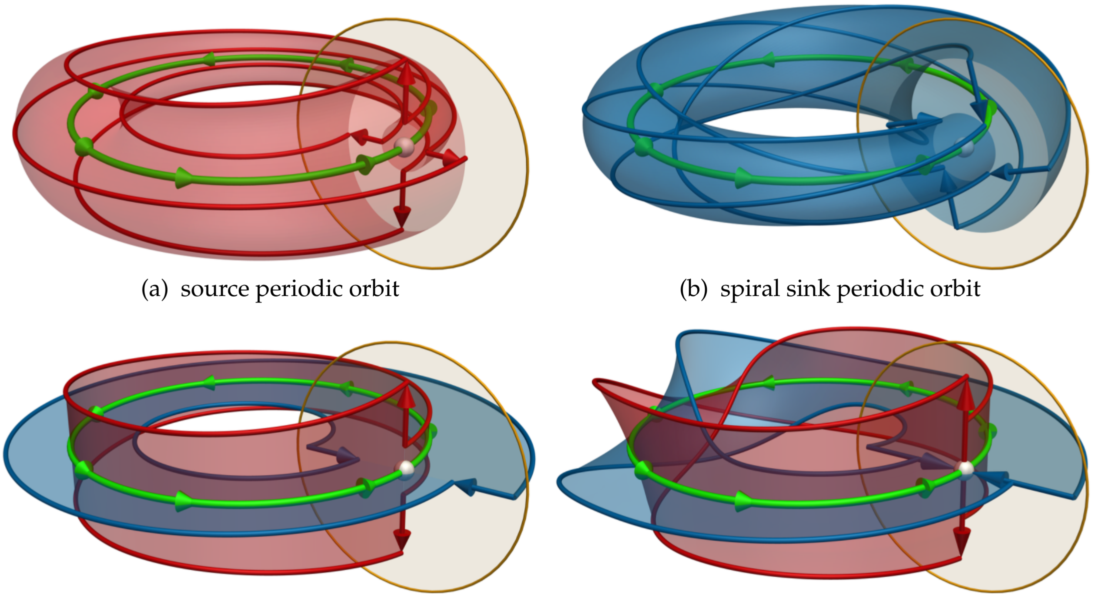
2.6.2 Periodic Orbits

An isolated closed streamline is called a *periodic orbit*. Since it is isolated, nearby streamlines converge to the periodic orbit in either forward or backward time. To analyze the behavior of such nearby streamlines, we consider a (n-1)-dimensional hyperplane Σ through a point $\mathbf{x}_p$ on the periodic orbit oriented transversally, i.e., not tangential, to the flow $\mathbf{u}(\mathbf{x}_p)$. On a neighborhood $\mathcal{U} \subset \Sigma$ around $\mathbf{x}_p$, called a Poincaré section, the Poincaré map $\mathbf{P}(\mathbf{x}) : \mathcal{U} \rightarrow \mathcal{U}$ maps a point $\mathbf{x}$ to the first-intersection point of the streamline seeded at $\mathbf{x}$ with $\mathcal{U}$. The neighborhood $\mathcal{U}$ shall be chosen small enough, such that $\mathbf{P}(\mathbf{x}_p)$=$\mathbf{x}_p$ and the periodic orbit only intersects $\mathcal{U}$ once. The value $\mathbf{P}(\mathbf{x})$ is undefined in case the streamline seeded at $\mathbf{x}$ does not reach $\mathcal{U}$ or leaves the domain.

Fixed points of $\mathbf{P}(\mathbf{x})$ correspond to periodic orbits, and the behavior of nearby streamlines is determined in first-order approximation by the eigenvalues $\mu_1, \ldots, \mu_{n-1}$ of $\nabla\mathbf{P}(\mathbf{x}_p)$. Since we only consider structurally stable periodic orbits, we may assume that no eigenvalue has complex absolute value one. Attracting and repelling behavior of a periodic orbit is determined by $|\mu_i| < 1$ (attracting) and $|\mu_i| > 1$ (repelling), respectively. A periodic orbit that has both attracting and repelling eigenvalues is called *saddle*-type. A complex-conjugate pair of eigenvalues with non-zero imaginary parts makes the periodic orbit *spiraling*. Finally, a pair of eigenvalues with zero imaginary parts and negative real parts is called *twisted*, since the corresponding stream surfaces have the topology of a Möbius strip. Note that the latter two cases can each only occur in pairs, since they correspond to a rotation of eigenvectors along the periodic orbits, i.e., they span a 2D rotation plane.

This classification scheme has been employed by Asimov [Asi93] for periodic orbits in 2D and 3D vector fields. Since in a 2D vector field, Poincaré sections are one-dimensional, there are only source- and sink-type periodic orbits. In 3D vector fields, source- and sink-type periodic orbits have spiral-type variants, which have corresponding complex eigenvalues in $\nabla\mathbf{P}$. Additionally, saddle-type periodic orbits exist, which further have a twisted variant in the case of negative real parts. The different types of 3D periodic orbits are shown in Figure 2.11.

The Poincaré map itself has been employed for enhancing visualizations of periodic orbits [LKG98], by embedding a visual representation of the 2D map within the 3D visualization of the flow. Tricoche et al. [TGS11] have proposed methods for the visualization of topological structures within area-preserving maps, where Poincaré maps of volume-preserving flows are a special case.



(a) source periodic orbit

(b) spiral sink periodic orbit

(c) saddle periodic orbit

(d) twisted saddle periodic orbit

Figure 2.11: Types of periodic orbits (green) in 3D vector fields, classified by localized behavior (red/blue arrows) in the Poincaré section (orange) around the fixed point (white sphere) of the Poincaré map. Periodic orbits of type source (a) and sink (b) exhibit 3D unstable (red) and 3D stable (blue) manifolds, respectively. Complex eigenvalues correspond to spiraling behavior of streamlines around the periodic orbit ((b), shown for type sink only). Saddle-type periodic orbits (c),(d) have pairs of a 2D stable and an 2D unstable manifolds, which in the case of negative real parts have the topology of a Möbius band, and are called twisted (d).

Numerical methods for computing periodic orbits often rely on differentials of the Poincaré map [Sim90], where Newton descend may be employed to find the exact location of a periodic orbit [Deu11]. Such methods are computationally expensive, and rely on the prior knowledge of a suitable Poincaré section, i.e., they typically only extract a subset of the set of all periodic orbits, or a single one. Noting the limitations of this approach, we employ it in Section 4.6.6 for demonstration of the different types of four-dimensional saddle periodic orbits.

By considering the entry and exit points of streamlines within cells, Wischgoll and Scheuermann [WS01; WS02] proposed an algorithm for finding all sink- and source-type periodic orbits in vector fields defined on 2D and 3D grids. An approach independent of an underlying grid for 2D vector fields was proposed by Theisel et al. [TWHS04b], which extends the 2D vector field to a 3D vector field with third component constant zero, and obtains periodic orbits from intersections of suitably seeded stream surfaces. Using ridges in the FTLE fields, Kasten et al. [KRRS14] proposed a method for finding 3D saddle-type periodic orbits. As a special case of recirculation surfaces, Wilde et al. [WRT18b] are able to extract a subset of the saddle-type periodic orbits by essentially considering a dense set of Poincaré sections. Note that saddle-type periodic orbits are notoriously difficult to extract, since they generally cannot be obtained by numerical integration due to the unstable manifolds present in either integration direction. They can be seen as prototype for the notion of hyperbolic trajectories in time-dependent vector fields (Section 2.7.3).

Initial candidates lines can be refined toward Periodic orbits using, for example, variational methods, which iteratively minimize local errors [LC04]. Machado et al. [MSE13] employ locally extracted initial candidates followed by a local refinement to extract bifurcation lines, which captures a special class of uniformly hyperbolic saddle-type periodic orbits, but typically is only able to obtain small parts, if any, of periodic orbits. Exploiting properties of the planar circular restricted three-body problem, Schlei et al. [SHTG14] used a multiple shooting method to extract periodic orbits. The initial candidate extraction in this approach was further improved by Tricoche et al. [TSH20], with application to spacecraft trajectory design.

2.6.3 Invariant Manifolds

The previously discussed critical points and periodic orbits exhibit sets of streamlines that converge toward them in either forward- or backward-time. These sets have the structure of a differential manifolds, and are called invariant manifolds [Wig90, Ch. 3]. Invariant manifolds that converge to the critical point or periodic orbit in forward-time are called stable, while invariant manifolds that converge in backward-time are called unstable. Here, stability refers to the behavior of streamlines under perturbation of the seed point. A perturbation within a stable manifold results in a streamline that still converges, while a perturbation within an unstable manifold causes the streamline to diverge. Note that the behavior of streamlines nearby, but not contained in, an invariant manifold is reversed: a streamline close to a stable manifold is repelled by it, while a streamline close to an unstable manifold is attracted toward it.

Invariant manifolds of critical points in linear vector fields are linear subspaces, spanned by the corresponding eigenvectors of the Jacobian (red and blue in Tables 2.1 and 2.2). In the case of a complex conjugate pair of eigenvectors, the real and imaginary parts taken separately as real vectors span the plane of rotation. For general nonlinear vector fields, we consider a small neighborhood around the critical point, where the linearization using the Jacobian (Equation 2.24) is valid. A stream manifold seeded at an offset within this neighborhood extends the linear part of the invariant manifold to

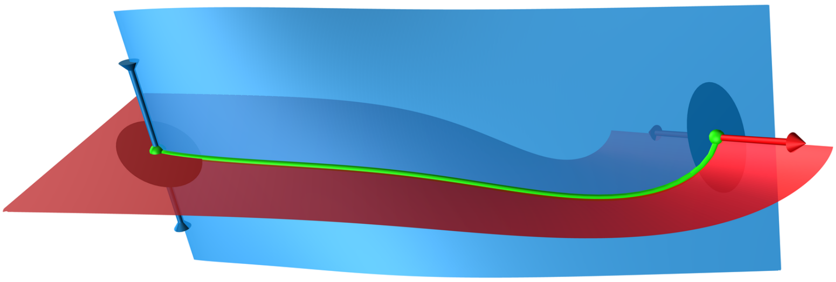


Figure 2.12: Example of a saddle connector (green line) in a steady 3D vector field that connects a 1:2 saddle-type critical point (left, green sphere) with a 2:1 saddle-type critical point (right, green sphere). This connection is the intersection curve of an unstable (red surface) and a stable (blue surface) invariant manifold, and thus a streamline.

the full nonlinear invariant manifold [Wig90, Sec. 3.2]. As discussed in the classification, the dimension of the invariant manifolds is determined by the number of positive (unstable) and negative (stable) real parts of eigenvalues. Source- and sink-type critical points have unstable and stable manifolds of codimension zero, which fill out the entire spatial domain of the vector field. Saddle-type critical points, on the other hand, posses invariant manifolds of codimension at least one, and thus separate the domain into regions, where all streamlines converge to the same pair of sink- and source-type critical points in forward and backward time.

For periodic orbits, invariant manifolds are obtained in a similar fashion [Wig90, Sec. 3.3]. In this case, linear segments of the invariant manifolds in the Poincaré map, i.e., within the Poincaré section, are extended to the nonlinear invariant manifolds using stream manifold integration. Again, only the saddle-type periodic orbits exhibit codimension-one separating manifolds.

2.6.4 Saddle Connectors

The intersection curve of two 2D invariant manifolds, i.e., stream surfaces, of two saddle-type critical points in a 3D steady vector field is called a saddle connector [TWHS03]. As an intersection of surfaces they are structurally stable, and, since they are stream surfaces, the intersection curve is a streamline (see Figure 2.12). We are going to see that saddle connectors in 4D steady vector fields can be lines or surfaces instead (Section 4.6.2), and their time-dependent equivalents are further going to be discussed in Section 5.4.5.

Theisel et al. [TWHS03] extract saddle connectors by geometrically computing the intersection of the 2D manifolds of all saddle-type critical points. A class of saddle connectors is captured by the concept of bifurcation lines [MSE13], but the algorithm proposed by the authors is typically not able to extract saddle connectors along their entire length. In dynamical systems literature, saddle connectors are called homoclinic and heteroclinic connections, and are related to chaos [Wig13, Sec. 2.5]. They are extracted by solving associated boundary value problems [Bey90; FD91], or by generalization of variational methods for periodic orbits [DL14], in this context.

2.6.5 Further Structures

A number of further topological structures haven been considered for the visualization of steady vector fields.

In Hamiltonian systems, quasiperiodic orbits are known to exist by the Kolmogorov–Arnold–Moser (KAM) theorem [Wig13, Sec. 2.7], which are confined to invariant tori (stream surfaces). Under perturbation, they become Cantor sets (measure zero sets), and are called Cantori. Peikert and Sadlo [PS09a] have considered invariant tori and Cantori for visualization in the context of vortex rings.

Including the boundary of the dataset, boundary switch points [DV99] on the boundaries of 2D steady vector fields have been proposed, which separate boundary outflow from boundary inflow. Scheuermann et al. [SHJK00] further considered interactively moving a bounded region around a dataset and visualizing the induced local topology. In 3D vector fields, the corresponding structures are boundary switch curves [WTHS04], which represent those locations on the boundary where the flow direction is tangential to the boundary surface.

On no-slip boundaries, i.e., surfaces where the velocity approaches zero, Kenwright et al. [Ken98; KHL99] proposed an extraction method for attachment and separation lines. These are defined as curves where the 2D wall shear stress, i.e., the derivative of velocity normal to the surface, is parallel to an eigenvector of its Jacobian. In 2D flows, where the boundaries are one-dimensional, they are represented by critical points in the 1D wall shear stress. Unsteady equivalents of attachment and separation lines in the space-time vector field are further a building block for time-dependent vector field topology ([MBES16], Section 2.7.3).

Considering critical points, where the Jacobian matrix is zero or has zero eigenvalues, the notion of higher-order critical points has been proposed for 2D vector fields [Sch+97; SKMR98]. While these are not structurally stable, they can be employed for topological simplification [TSH00; TSH01a], where clusters of critical points are replaced by single higher-order critical points. These notions, as well as the topological simplification, has been extended to 3D by Weinkauf et al. [Wei+05].

2.7 Time-Dependent Vector Field Topology

There exist several different ways to extend vector field topology of steady vector fields, as discussed in the previous section, to time-dependent vector fields.

A natural extension is tracking the evolution of the frozen-time (Eulerian) steady VFT over time. Tricoche et al. [TSH01b; TWSH02] proposed tracking 2D VFT using linear interpolation between time steps. This approach was extended to the 3D case by Garth et al. [GTS04]. Theisel et al. [TWHS04a; TWHS05] proposed streamline- and pathline-based approaches for 2D time-dependent vector fields. In their streamline-based approach, critical points and periodic orbits are tracked using the feature flow field [TS03], while their pathline-based approach extracts local properties of pathlines.

These methods are, however, dependent on the frame of reference that the vector field is observed in. This is especially problematic in the presence of an ambient flow, in which case the lab frame of reference (e.g., the observer attached to an obstacle in the flow) cannot capture any meaningful structure. Based on the Helmholtz–Hodge decomposition, Wiebel et al. [WGS07], and Bhatia et al. [BPB14; BPKB14] proposed to visualize topological features in the internal frame of reference obtained from this decomposition. In a similar fashion, Baeza Rojo and Günther [BG20] proposed to employ steady VFT in a displacement-invariant frame of reference to obtain separating structures of the time-dependent flow.

Note that these methods are local in time, i.e., they attempt to reduce the time-dependent dynamics to a single time step. Therefore, they cannot capture the time-dependent behavior of massless particles, such as observed by pathlines, streaklines or timelines, in vector fields with general time-dependence. Methods based on the full time-dependent dynamics are also called Lagrangian, and we focus on these for the remainder of this section. Throughout this thesis, by *time-dependent vector field topology* we mean a Lagrangian generalization of steady VFT (see Chapter 5).

2.7.1 FTLE and Lagrangian Coherent Structures

Recall that the flow of a time-dependent vector field is given by its flow map $\boldsymbol{\phi}_{t_0}^{T}(\mathbf{x})$, which maps a seed point $\mathbf{x}$ at time t_0 to its final position after advection time T. Expo-

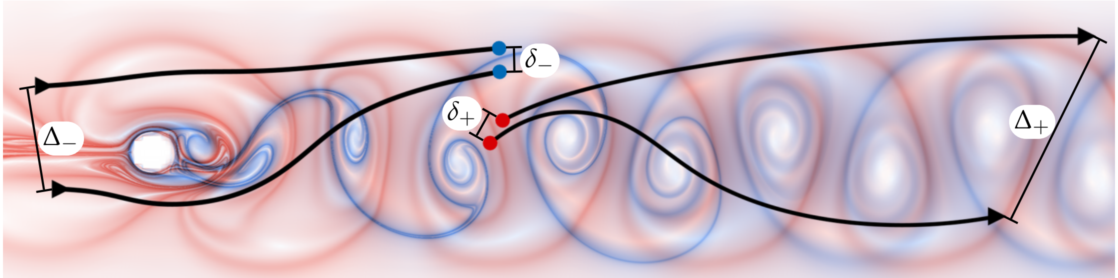


Figure 2.13: Forward- and backward-time FTLE (mapped to red/blue color) of the 2D Cylinder Flow (Section 5.6.1) at t_0=12.5 s with T=±1.5 s. Ridges in the forward- and backward-time FTLE fields represent repelling and attracting LCS. Pathlines (black lines) seeded on opposing sides of FTLE ridges (blue circles: backward-time pathlines seeded across attracting LCS, red circles: forward-time pathlines seeded across repelling LCS) show growth ($\Delta_\pm$) of initial perturbations ($\delta_\pm$) in the respective integration directions. The FTLE represents the limit case $\delta_\pm \to 0$.

nential separation, the main organizing structure of the flow, can be obtained from the FTLE field defined as

$$\varsigma_{t_0}^{T}(\mathbf{x}) = \frac{1}{|T|} \ln \sqrt{\mu_{\max}\left(\left(\nabla \boldsymbol{\phi}_{t_0}^{T}(\mathbf{x})\right)^{\top} \nabla \boldsymbol{\phi}_{t_0}^{T}(\mathbf{x})\right)}, \tag{2.25}$$

where $\mu_{\max}(\mathbf{M})$ denotes the largest eigenvalue of the real symmetric matrix $\mathbf{M}$. For an n-dimensional flow, the FTLE is a spectrum of n exponents, but the largest is typically used as a synonym. A subset of the ridges in $\varsigma_{t_0}^{T}(\mathbf{x})$ for negative advection time represent attracting Lagrangian coherent structures (LCS), while they represent repelling LCS for positive advection time [Hal01; SLM05]. Generally, particles separate exponentially in forward time, when they are close to repelling LCS, and they separate exponentially in backward time, when they are close to attracting LCS (Figure 2.13).

A variety of competing concepts for the definition and computation of LCS has been proposed [Had+17; OHH15]. For the visualization of unsteady vector fields, the FTLE-based approach has been successfully employed [Gar+09; SP09], which we are going to use as baseline in this thesis. These methods rely on computing gradients of the flow map, which generally requires very fine sampling near the LCS. To counter this, adaptive methods have been proposed [GGTH07; SP07; SRP11]. Üffinger et al. [Üff+12] further proposed a higher-order computation of the flow map, and Kuhn et al. [Kuh+14] obtained LCS by timeline tracking, which does not require the flow map gradient. For a robust extraction of FTLE ridges for long integration times, Wilde et al. [WRT18a] proposed to further incorporate the FTLE at shorter integration times.

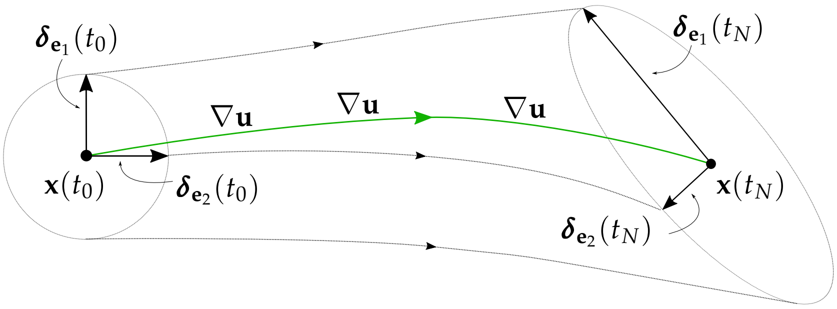

$\boldsymbol{\delta}_{\mathbf{e}_1}(t_0)$
$\nabla\mathbf{u}$
$\nabla\mathbf{u}$
$\nabla\mathbf{u}$
$\boldsymbol{\delta}_{\mathbf{e}_1}(t_N)$
$\mathbf{x}(t_0)$
$\boldsymbol{\delta}_{\mathbf{e}_2}(t_0)$
$\mathbf{x}(t_N)$
$\boldsymbol{\delta}_{\mathbf{e}_2}(t_N)$

Figure 2.14: The localized flow of a 2D vector field along a pathline $\mathbf{x}(t)$ (green) describes the evolution of infinitesimal perturbations $\boldsymbol{\delta}_{\mathbf{e}_1}(t_0) = \mathbf{e}_1, \boldsymbol{\delta}_{\mathbf{e}_2}(t_0) = \mathbf{e}_2$ (left ellipse) around $\mathbf{x}(t)$ over a finite time interval $[t_0, t_N]$. In first-order approximation, the evolution is obtained from the Jacobian $\nabla\mathbf{u}$ along the pathline. The integrated perturbations $\boldsymbol{\delta}_{\mathbf{e}_1}(t), \boldsymbol{\delta}_{\mathbf{e}_2}(t)$ form the columns of the fundamental solution matrix $\mathbf{X}_{t_0}(t)$ of the localized flow (Equation 2.26).

Instead of using a computational grid to obtain the necessary flow map gradients, Kasten et al. [Kas+09] integrated the localized flow along trajectories, which we discuss in the following section.

2.7.2 Localized Finite-Time Lyapunov Exponents

While the FTLE field conceptually describes separation of neighboring particles, we now consider the (localized) finite-time Lyapunov exponents of single pathlines. Local separation and attraction in an infinitesimal neighborhood (Figure 2.14) of a pathline $\mathbf{x}(t)$, with $\mathbf{x}(t_0) = \mathbf{x}_0$, over a time interval $[t_0, t_N]$, is described by the localized flow

$$\frac{d}{dt}\boldsymbol{\delta}_{\mathbf{x}}(t) = \nabla\mathbf{u}(\mathbf{x}(t), t)\boldsymbol{\delta}_{\mathbf{x}}(t). \tag{2.26}$$

This linear ODE describes the evolution of an infinitesimal neighborhood $\boldsymbol{\delta}_{\mathbf{x}}(t)$ along $\mathbf{x}(t)$, which is captured entirely by its solution with initial condition $\mathbf{X}_{t_0}(t_0) = \mathbb{I}$. We call this solution the fundamental solution matrix $\mathbf{X}_{t_0}(t)$. In previous work, it has been used to compute a localized variant of the FTLE [Kas+09], as well as streaklines as tangent curves [WHT12; WT10]. Using the singular value decomposition

$$\mathbf{X}_{t_0}(t) = \mathbf{B}_{t_0}(t)e^{\boldsymbol{\Sigma}_{t_0}(t)}\mathbf{R}_{t_0}(t)^\top, \tag{2.27}$$

with orthogonal matrices $\mathbf{B}_{t_0}(t)$, $\mathbf{R}_{t_0}(t)$, and a diagonal matrix $\boldsymbol{\Sigma}_{t_0}(t)$ with diagonal entries $\sigma^1_{t_0}(t) \geq \sigma^2_{t_0}(t)\left[\geq \sigma^3_{t_0}(t)\right]$, the localized finite-time Lyapunov exponent spectrum is obtained as

$$\lambda^i_{t_0}(t) = \frac{1}{|t - t_0|}\sigma^i_{t_0}(t). \tag{2.28}$$

Note that the finite-time Lyapunov exponent is often referred to as the largest one, i.e., $\varsigma_{t_0}^{t-t_0} = \lambda_{t_0}^1(t)$, which measures exponential separation from time t_0 to t. However, since integration can be reversed, we have $\mathbf{X}_t(t_0) = \mathbf{X}_{t_0}(t)^{-1}$. Thus, we also obtain exponential attraction from the smallest Lyapunov exponent $\lambda_{t_0}^n(t)$, i.e., exponential separation in backward direction from time t to t_0. This fact can also be exploited for computing both forward-time and backward-time FTLE fields from a single computation using a grid of pathlines [HS11].

In linear flows, $\nabla \mathbf{u}$ is a constant matrix, and thus Equation 2.26 is identical for all pathlines $\mathbf{x}(t)$. In this case, the FTLE $\varsigma_{t_0}^T(\mathbf{x})$ is thus a constant value for all $\mathbf{x}$, and is unable to reveal flow structure, i.e., the FTLE relies on heterogenity of the phase space.

2.7.3 Hyperbolic Trajectories and Streak-Based Topology

The FTLE-based definition of LCS has the disadvantage that it is implicit, i.e., it has to be combined with a geometric ridge extraction in order to be used for further calculation. Even after ridge extraction, their temporal evolution still needs to be established by a tracking approach. Hyperbolic trajectories, which are pathlines that have stable and unstable manifolds representing the LCS, provide a solution to this problem.

Haller [Hal00] gives a definition for hyperbolic trajectories as found in typical two-dimensional fluid flows. This definition requires instantaneous hyperbolicity, i.e., the Jacobian $\nabla \mathbf{u}(\mathbf{x}(t), t)$ possessing eigenvalues of alternating signs along the trajectory $\mathbf{x}(t)$, as well as further conditions on the eigenvalues and eigenvectors of the Jacobian, which the author calls uniform hyperbolicity. As a necessary condition, maximum time spent in a instantaneously hyperbolic region is derived.

Sadlo and Weiskopf [SW10] found that this hyperbolicity time criterion is numerically unstable, and proposed to extract seeding points by intersection of forward- and backward-time FTLE ridges in single time steps. From these, the hyperbolic trajectories are obtained by pathline integration. Subsequently, the LCS are obtained by seeding generalized streaklines at an offset along the hyperbolic trajectories. Since, computing a streakline from the hyperbolic trajectory itself would result in a pathline rather than a streakline, Sadlo and Weiskopf [SW10] interpret them as degenerate streaklines. Thus, this concept is in analogy to steady vector field topology, where saddle-type critical points can be regarded as degenerate streamlines. The authors call their generalization *streak-based topology*. Üffinger et al. [ÜSE13] extended this approach to three-dimensional flows, where hyperbolic path surfaces instead of hyperbolic trajectories are considered. Similarly, path surfaces are seeded from intersection curves of FTLE ridge surface. A

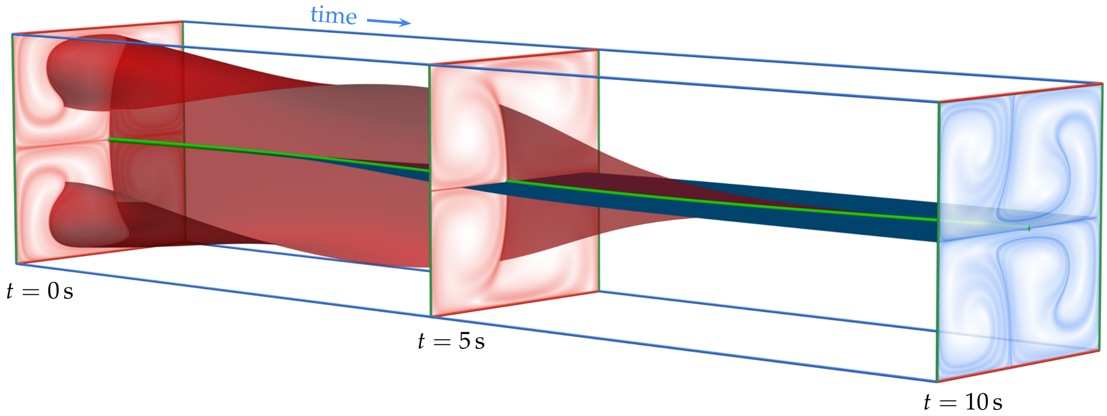


Figure 2.15: Streak topology of the 2D modified double gyre [WWRT21] with ground truth hyperbolic trajectory (green), in space-time view, restricted to the time interval $[0\,\mathrm{s}, 10\,\mathrm{s}]$ without periodic boundary conditions, and spatial domain $[0,2]\times[-1,1]$. Repelling streak manifold (red surface) of the hyperbolic trajectory coincides with repelling LCS, represented by ridges in the forward-time FTLE field (red, shown at t=0 s and t=5 s, with advection times T=10 s and T=5 s), and attracting streak manifold (blue surface) coincides with attracting LCS, represented by ridges in the backward-time FTLE field (blue, shown at t=10 s, with advection time T=−10 s).

2D example is shown in Figure 2.15. Note that the coloring scheme of steady vector field topology (blue stable, red unstable) is reversed in order to reflect attracting (blue) and separating (red) behavior of the manifold, as the notion of convergence, and thus stability, is less relevant in the finite-time setting.

The pathline integration involved in these approaches is, however, numerically unstable, due to the repelling manifolds involved in either time direction. Motivated by Haller's definition of a hyperbolic trajectory, Machado et al. [MBES16] extract bifurcation lines in the 3D space-time vector field. This approach avoids numerical integration altogether, and is based on local refinement of space-time parallel vectors solutions $\bar{\mathbf{u}} \parallel (\nabla\bar{\mathbf{u}})\bar{\mathbf{u}}$ towards the nearest pathline. Following the discussion in Section 2.5.5, this allows to reinterpret the method as tracking saddle-type critical points in the Galilean-invariant frame of reference defined by the feature flow field of $\mathbf{u}$ to obtain candidate lines for refinement. In flows with no-slip boundaries, the authors further propose to extract space-time separation and attachment lines to obtain additional candidate lines. As we are going to see in Chapter 5, this local refinement does not always, if at all, converge to the nearest hyperbolic trajectory.

Recently, Wolligandt et al. [WWRT21] presented a 2D time-periodic vector field with ground-truth hyperbolic trajectory. The authors also show that the approach by Machado et al. [MBES16] is unreliable in certain cases.

A dynamical systems definition of hyperbolic trajectories is given by Ide et al. [ISW02], who define them as pathlines that possess an exponential dichotomy [Cop78]. Equivalently, a hyperbolic trajectory is a pathline with only non-zero finite-time Lyapunov exponents that possess invariant manifolds. Because this definition captures most trajectories, the authors further define distinguished hyperbolic trajectories (DHTs) as the topologically relevant hyperbolic trajectories (see Section 5.1 for a detailed definition). This definition includes the case of codimension zero invariant manifolds, i.e., source- and sink-type DHTs, similar to the definition by Bujack et al. [BDZG19; Buj+19]. Ide et al. further propose an algorithm for refinement of paths of critical points in the lab frame of reference toward DHTs based on a Fourier transform, implying periodicity in time. Mancho et al. [MSW04] propose a modification of this algorithm without the time-periodicity assumption. Existence and uniqueness of DHTs is proven by Ju et al. [JSW03], which involves a fixed-point iteration. This technique is employed for the extraction of DHTs in 3D by Branicki and Wiggins [BW09]. We are going to employ an extension of this approach in both the 2D and 3D cases in Chapter 5.

In the context of DHTs, the FTLE was used for verification of the associated invariant manifolds by Branicki and Wiggins [BW10]. Mancho et al. [MWCM13] introduced Lagrangian descriptors, which are based on integration of scalar quantities, such as velocity magnitude or pathline curvature, along pathlines in forward and backward time. Similarly to the FTLE, these also reveal invariant manifolds of DHTs, but do not suffer from the same limitations, e.g., they are applicable to linear flows, and are shown to require shorter advection times for convergence. Nevertheless, we are going to use the FTLE for validation throughout this thesis, since its structures, which directly relate to separation of pathlines, are easier to interpret.

2.8 Higher-Dimensional Visualization

In the preceding part of this chapter, two- and three-dimensional fields were the main focus. Only few works so far considered the visualization higher-dimensional fields and geometry, which we outline in this section.

Visualization of 4D geometry has been first explored in an early approach by Noll [Nol67], where an automatic plotter was used to render frames of a two-dimensional movie showing projections of rotating four-dimensional geometry. This technique was limited to wireframe depictions, and has later been implemented on a Silicon Graphics workstation [Hol91]. Subsequently, lighting models in four dimensions were proposed [HC93; HH92], as well as an interactive tool for visualizing four-

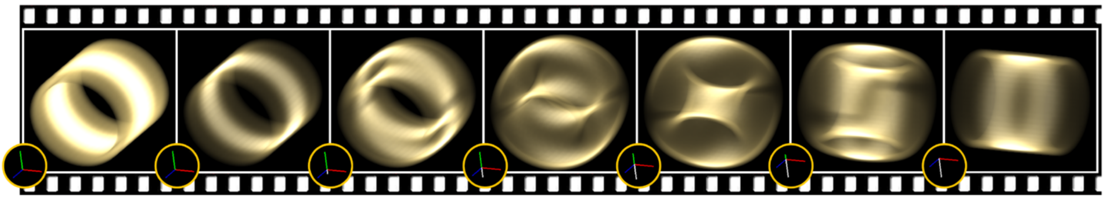


Figure 2.16: Flat torus (2-manifold) embedded in 4D Euclidean space, rendered using 4D lighting model and volumetric frame buffer. Rotation in the (z, w)-plane in 4D space (axis icons) reveals its internal structure. Image © 2009 IEEE, reproduced, with permission, from [CFHH09].

dimensional surfaces [HIM99]. Stereographic projections have been employed as an alternative for the visualization of contours of complex-valued bivariate functions [WB96], where the space $\mathbb{C} \times \mathbb{C}$ is identified with four-dimensional Euclidean space $\mathbb{R}^4$. Instead of projecting from 4D to a 2D canvas, Chu et al. [CFHH09] utilized a three-dimensional frame buffer for volume rendering of 3D projections of 4D surfaces and volumes (Figure 2.16). In this context, the problem of self-intersections has been addressed with higher-dimensional halos [Wan+13]. We are going to employ projections of 4D geometry in Chapter 4 for the visualization of 4D steady vector field topology, where we further use an approach similar to generalized halos for reducing ambiguities in our glyph representations of critical points.

Higher-dimensional scalar fields have been visualized using localized approaches, by showing multiple 2D slices containing same point of interest [Wij93], using projections onto a dense sequence of 2D subspace [Asi85], as well as 2D and 3D projections around a point of interest [PBK10]. For indirect visualization, Bhaniramka et al. [BWC04] proposed a generic algorithm for extracting codimension-one isocontours in scalar fields of arbitrary dimension, which is a generalization of the marching cubes algorithm [LC87].

For visualizing the relationship between multivariate data and geometry, star coordinates [Kan00] and parallel coordinates [Ins85] have been used. The latter has been employed by Wegenkittl et al. [WLG97] and Grottel et al. [GHWG14] for visualizing trajectories in higher-dimensional dynamical systems. Amirkhanov et al. [Ami+19] proposed to employ an animated transition between 2D, 3D, and 4D projections for the visualization of trajectories in a four-dimensional flow. Nonlinear projections for higher-dimensional dynamical systems have been proposed by Bartolovic et al. [BGG20], which preserve certain properties of the vector field by solving an optimization problem. This technique can be regarded as a specialized dimensionality reduction method [NA18; VPV+09], which are typically employed for sets of discrete data points.

The special case of inertial particles was treated by Günther et al. [GG17; GT15; GT16a; GT16b; Gün16], where the n-dimensional phase space of massless particles in a flow is extended by another n dimensions corresponding to momentum. Since the $2n$ dimensions correspond to different physical units, mostly projections onto the n-dimensional subspaces corresponding to location or momentum are used. Baeza Rojo et al. [BGG18] proposed coordinated views, instead, to show the influence of varying particle seeding positions, velocities and sizes at once. These works consider models with underlying n-dimensional vector fields based on response time that exhibit global attracting manifolds. For generic second-order dynamical systems, the concept of LCS has been extended [Sag+17; SBHH15] and applied to n-body problems. Both Sagristà et al. [Sag+17] and Günther and Theisel [GT16b] used dimensional stacking for visualization of four-dimensional phase space.

In the following chapters of this thesis we present several approaches to the visualization of higher-dimensional fields. Part I presents a method for the definition and extraction of local features in scalar and vector of arbitrary dimension. In Part II we turn to the topology of four-dimensional and time-dependent vector fields. For 4D vector field topology, we present a projection approach tailored to topological skeletons. Since 3D time-dependent vector fields posses 4D space-time domains, the latter is also a higher-dimensional problem, and we are going to be able to transfer insights from the preceding chapters there.

Part I

Local Features

3 The Dependent Vectors Operator

One of the striking advantages of feature extraction is that it is often parameter-free. In that sense, features are (typically lower-dimensional) subsets of the domain that exhibit special characteristics, which do not depend on user-defined parameters. A wide range of feature definitions, along with respective extraction techniques, is in use in various fields, and for various applications (see Section 2.5).

Since the projection of higher-dimensional objects onto 3D space usually leads to self-intersection and is intrinsically harder to interpret, the extraction of meaningful features from higher-dimensional spaces is even more important for their visualization. Features extracted from higher-dimensional vector fields and scalar fields have in common that one can no longer only consider line-type or codimension one features, such as extracted by the parallel vectors operator (Section 2.5.1). For example, vortex core manifolds, generalized vortex core lines, are surfaces in 4D vector fields, and lines or volumes in 5D vector fields.

In this chapter, we generalize the PV operator to fields with arbitrary dimension larger than one, leading to the dependent vectors (DV) operator. This operator extracts manifolds, where an n-dimensional vector field together with a set of k (derived) nD vector fields is linearly dependent. In the 2D and 3D cases, our DV operator is identical to the PV operator. We show its utility by demonstrating how it is used to define vortex core manifolds, bifurcation manifolds, and height ridge manifolds in higher dimensions. For the extraction of the respective DV solutions (i.e., the manifolds with arbitrary dimension), we present a generic, simple, and effective algorithm, and demonstrate it at these cases.

Our contributions include:

- generalization of the PV operator to arbitrary dimension,
- a generic algorithm for extracting the resulting manifolds, and
- application of our approach to generalized vortex core manifolds, bifurcation manifolds, as well as ridge and valley manifolds.

3.1 Operator and Feature Definitions

We consider an n-dimensional vector field $\mathbf{u}(\mathbf{x})$ and a set of k vector fields $\mathbf{w}_1(\mathbf{x}), \dots, \mathbf{w}_k(\mathbf{x})$, all defined on the same domain $\Omega \subset \mathbb{R}^n$, with $\mathbf{u}(\mathbf{x}), \mathbf{w}_i(\mathbf{x}) \in \mathbb{R}^n$ (note that we often write, e.g., $\mathbf{u}$ for $\mathbf{u}(\mathbf{x})$ in our notation, were not ambiguous). For a fixed $\mathbf{x} \in \mathbb{R}^n$, we call the vectors $\mathbf{u}(\mathbf{x}), \mathbf{w}_1(\mathbf{x}), \dots, \mathbf{w}_k(\mathbf{x})$ linearly dependent, if there exist scalars $c_1(\mathbf{x}), \dots, c_k(\mathbf{x}) \in \mathbb{R}$ such that

$$\mathbf{u}(\mathbf{x}) = c_1(\mathbf{x})\mathbf{w}_1(\mathbf{x}) + \dots + c_k(\mathbf{x})\mathbf{w}_k(\mathbf{x})\,. \tag{3.1}$$

Here, we assume that the vectors $\mathbf{w}_1(\mathbf{x}), \dots, \mathbf{w}_k(\mathbf{x})$ are linearly independent and none of the vectors is zero. This notion directly generalizes that of two vectors being parallel, by which we mean that there is a $c(\mathbf{x}) \in \mathbb{R}$, such that $\mathbf{u}(\mathbf{x}) = c(\mathbf{x})\mathbf{w}(\mathbf{x})$. Equation 3.1 is a set of n equations with unknowns $\mathbf{x} \in \mathbb{R}^n$, and $c_1, \dots, c_k \in \mathbb{R}$, such that its solutions are k-dimensional manifolds. We define the dependent vectors operator as the operator that, given nD vector fields $\mathbf{u}, \mathbf{w}_1, \dots, \mathbf{w}_k$, returns the set $\mathcal{D}$ of solution points of Equation 3.1.

3.1.1 Wedge Product

Two three-dimensional vectors $\mathbf{u}, \mathbf{w} \in \mathbb{R}^3$ are parallel if and only if their cross product $\mathbf{u} \times \mathbf{w}$ is the zero vector. The appropriate notion of such a cross product in arbitrary dimension and for an arbitrary number of vectors is that of the *wedge product*, which we introduce in the following. Just like the norm of the 3D cross product coincides with the area of the parallelogram spanned by the two vectors, the norm of the wedge product $\mathbf{u} \wedge \mathbf{w}_1 \wedge \dots \wedge \mathbf{w}_k$ of the $k+1$ vectors $\mathbf{u}, \mathbf{w}_1, \dots, \mathbf{w}_k \in \mathbb{R}^n$ coincides with the (k+1)-dimensional volume spanned in n-dimensional space. As such, the wedge product is zero if and only if the vectors $\mathbf{u}, \mathbf{w}_1, \dots, \mathbf{w}_k$ are linearly dependent, i.e., they lie in a linear subspace of dimension lower than k+1. Note, however, that two non-parallel four-dimensional vectors $\mathbf{u}, \mathbf{w} \in \mathbb{R}^4$ exhibit a full two-dimensional linear subspace of vectors that are orthogonal to $\mathbf{u}$ and $\mathbf{w}$. Therefore, the wedge product of n-dimensional vectors will in general no longer be again an n-dimensional vector.

If the vectors $\mathbf{u}, \mathbf{w}_1, \dots, \mathbf{w}_k \in \mathbb{R}^n$ with $k + 1 < n$ are linearly independent, i.e., Equation 3.1 has no solution, they can be extended to a set of n linearly independent vectors by "filling in" appropriately chosen vectors from a basis of $\mathbb{R}^n$, such as the standard basis $\mathcal{E} := \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$. If a choice of $m = n - k - 1$ vectors $\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_m}$ does not extend the vectors to a basis, we have

$$\det(\mathbf{u}, \mathbf{w}_1, \ldots, \mathbf{w}_k, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_m}) = 0\,. \tag{3.2}$$

The vectors $\mathbf{u}, \mathbf{w}_1, \ldots, \mathbf{w}_k$ are thus linearly dependent, if and only if for all choices $i_1, \ldots, i_m \in \{1, \ldots, n\}$, Equation 3.2 holds. By symmetry, it suffices to consider the $l := \binom{n}{k}$ choices $1 \leq i_1 < \cdots < i_m \leq n$. While usually the wedge product is defined as an antisymmetric order-m tensor consisting of terms of Equation 3.2 indexed by $i_1, \ldots, i_m$, we are only interested in its l degrees of freedom. Therefore, we choose coordinates in $\mathbb{R}^l$ by enumerating the subsets $\mathcal{E}_1, \ldots, \mathcal{E}_l \subseteq \mathcal{E}$ of the standard basis lexicographically, and define

$$\mathbf{u} \wedge \mathbf{w}_1 \wedge \cdots \wedge \mathbf{w}_k := (\det(\mathbf{u}, \mathbf{w}_1, \ldots, \mathbf{w}_k, \mathcal{E}_i))_{i=1}^{l} \in \mathbb{R}^l \tag{3.3}$$

as the wedge product of the vectors $\mathbf{u}, \mathbf{w}_1, \ldots, \mathbf{w}_k$, i.e., our wedge product is an l-dimensional vector. The wedge product is zero whenever the vectors are linearly dependent, and we will treat this choice of coordinates as the definition of the wedge product for the remainder of this chapter.

Note that this is not the definition of the wedge product used in common mathematics literature [Lou01, Ch. 3], where Equation 3.3 can be understood, instead, as a choice of coordinates in the dual space induced by the Hodge star operator. We deliberately choose this definition, such that it is consistent with the three-dimensional case, which we aim to generalize. By definition, for two 3D vectors $\mathbf{u}, \mathbf{w} \in \mathbb{R}^3$, we have

$$\mathbf{u} \wedge \mathbf{w} = \begin{pmatrix} \det(\mathbf{u}, \mathbf{w}, \mathbf{e}_1) \\ \det(\mathbf{u}, \mathbf{w}, \mathbf{e}_2) \\ \det(\mathbf{u}, \mathbf{w}, \mathbf{e}_3) \end{pmatrix} = \mathbf{u} \times \mathbf{w}\,, \tag{3.4}$$

i.e., the wedge product of two 3D vectors is identical to their 3D cross product. It can thus be seen that the choice of a basis $\mathcal{E}$ is independent of the vector fields $\mathbf{u}, \mathbf{w}_1, \ldots, \mathbf{w}_k$. Furthermore, in the cases $k = n - 2$, the wedge product maps a set of $n - 1$ n-dimensional vectors to an n-dimensional vector, which is orthogonal to each of them, thus providing a generalized cross product. In the cases $k = n - 1$, the wedge product is scalar-valued, with $\mathbf{u} \wedge \mathbf{w}_1 \wedge \cdots \wedge \mathbf{w}_{n-1} = \det(\mathbf{u}, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1})$.

Using the notion of the wedge product, the solution set $\mathcal{D}$ of the dependent vectors operator can be written as

$$\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{u}(\mathbf{x}) \wedge \mathbf{w}_1(\mathbf{x}) \wedge \cdots \wedge \mathbf{w}_k(\mathbf{x}) = \mathbf{0}\}\,, \tag{3.5}$$

keeping in mind that the vectors $\mathbf{w}_1(\mathbf{x}), \ldots, \mathbf{w}_k(\mathbf{x})$ are assumed to be linearly independent. The set $\mathcal{D}$ is thus the intersection of l zero-level sets of scalar functions, each of which are closed, and as such is closed as well.

3.1.2 Height Ridge Manifolds

A k-dimensional height ridge of an nD scalar field f is the set of points, where f has a local maximum in $n-k$ directions. According to the definition of Eberly et al. [Ebe+94], these are the points $\mathbf{x}$, where

$$\boldsymbol{\eta}_1 \cdot \nabla f = \cdots = \boldsymbol{\eta}_{n-k} \cdot \nabla f = 0\,, \tag{3.6}$$

$$\mu_1, \ldots, \mu_{n-k} < 0\,, \tag{3.7}$$

where $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_n$ denote the eigenvectors of the Hessian $\nabla\nabla f$ at $\mathbf{x}$, with respective real eigenvalues, in ascending order, $\mu_1 \leq \cdots \leq \mu_n$.

In other words, the definition requires that the $n-k$ smallest eigenvalues are negative, and that the respective second directional derivatives are negative, i.e., that the directional derivatives of f along the corresponding eigenvectors are zero. Valleys, the opposite of ridges, are obtained by height ridge extraction from $-f$.

The condition of Equation 3.6 is equivalent to requiring that the gradient ∇f lies in the linear subspace spanned by the eigenvectors $\boldsymbol{\eta}_{n-k+1}, \ldots, \boldsymbol{\eta}_n$. That is, for a fixed $\mathbf{x} \in \mathbb{R}^n$, there exist $c_{n-k+1}, \ldots, c_n \in \mathbb{R}$, such that

$$\nabla f(\mathbf{x}) = c_{n-k+1}(\mathbf{x})\boldsymbol{\eta}_{n-k+1}(\mathbf{x}) + \cdots + c_n(\mathbf{x})\boldsymbol{\eta}_n(\mathbf{x})\,. \tag{3.8}$$

Thus, height ridges can be expressed using the dependent vectors operator with $\mathbf{u} = \nabla f$ and $\mathbf{w}_i = \boldsymbol{\eta}_{n-k+i}$, $i = 1, \ldots, k$, where the solution set $\mathcal{D}$ is subsequently filtered according to Equation 3.7.

3.1.3 Vortex Core Manifolds

By the formulation of Sujudi and Haimes [SH95], a vortex core line in a 3D vector field is the set of points, where the flow direction $\mathbf{u}$ is governed by the real eigenvector of the

Jacobian $\nabla\mathbf{u}$, while the remaining two eigenvectors are complex-valued. The real and imaginary parts of the pair of complex-conjugate eigenvectors span a two-dimensional plane of rotation in 3D space, within which the reduced flow $\mathbf{u} - \mathbf{u}(\mathbf{x})$ rotates. The Sujudi–Haimes criterion can be reformulated using the PV operator by requiring that $\mathbf{u}$ is parallel to the real eigenvector and $\nabla\mathbf{u}$ has a pair of complex eigenvalues.

A rotation in n-dimensional space is defined by one or more two-dimensional planes of rotation. Note that a plane of rotation in n-space leaves an (n-2)-dimensional subspace invariant, which happens to be one-dimensional in 3D, and therefore the notion of a *rotation axis* is in general not available in higher dimensions. For $n \geq 4$, there exist rotations which simultaneously rotate in more than one plane of rotation. We call such rotations *r-rotations*, where r is the number of simultaneous rotation planes. The case $r = 1$ is also called a *simple rotation*.

Accordingly, an *r-vortex core manifold* in an n-dimensional vector field is a (n-2r)-manifold, within which the flow is mainly governed by its $k = n - 2r$ non-rotational directions. Following the 3D formulation, points that belong to such a manifold are defined by the flow direction $\mathbf{u}$ lying in the linear space spanned by the k real eigenvectors of $\nabla\mathbf{u}$. Thus, r-vortex core manifolds can be reformulated using the DV operator by defining $\mathbf{w}_1, \ldots, \mathbf{w}_k$ to be the k real eigenvectors of $\nabla\mathbf{u}$. Regions that do not have exactly k *real* eigenvectors are omitted (filtered).

Since their definition depends on two-dimensional planes of rotation, different kinds of vortex core manifolds exist, depending on the dimension n of the surrounding space. When n is even, $(n/2)$-vortex core manifolds are zero-dimensional and correspond to critical points of $\mathbf{u}$. Furthermore, vortex core manifolds need to be of even dimension. Vice versa, if n is odd, all vortex core manifolds need to be of odd dimension. For example, in 4-space, there are zero-dimensional 2-vortex core manifolds, and 2-dimensional 1-vortex core manifolds, but no 1-dimensional vortex core lines. Vortex core lines exist, e.g., in 5-space as 2-vortex core manifolds.

3.1.4 Bifurcation Manifolds

For 3D flows, Perry and Chong [PC87] proposed the notion of *bifurcation lines*, which are streamlines that exhibit an attracting and a repelling two-dimensional manifold of streamlines. Streamlines on the attracting manifold locally converge toward the bifurcation line, while streamlines on the repelling manifold converge to the bifurcation line in reverse time. Bifurcation lines thus locally separate the flow. According to Roth [Rot00], bifurcation lines may be obtained by a modification of the Sujudi–Haimes criterion, where instead of regions of the flow that have a complex plane of rotation one seeks those regions that have only real eigenvalues, and their signs are alternating.

To generalize this notion to arbitrary dimensions, we require that the Jacobian $\nabla \mathbf{u}$ has real eigenvalues $\mu_1 \leq \cdots \leq \mu_n$. A point $\mathbf{x}$ lies on a *bifurcation manifold*, if the flow $\mathbf{u}(\mathbf{x})$ lies in the (n-2)-dimensional space spanned by the eigenvectors $\boldsymbol{\eta}_2, \ldots, \boldsymbol{\eta}_{n-1}$ belonging to the medium eigenvalues and $\mu_1 \mu_n < 0$. Such a bifurcation manifold then exhibits attracting behavior in direction of the minor eigenvector $\boldsymbol{\eta}_1$, and repelling behavior in direction of the major eigenvector $\boldsymbol{\eta}_n$. Both eigenvectors define (n-1)-dimensional manifolds of streamlines, since integration adds one dimension to the (n-2)-dimensional bifurcation manifold. Similarly to vortex core manifolds, bifurcation manifolds can be obtained by the DV operator by rejecting regions of the raw solutions where the Jacobian has complex eigenvalues.

3.2 Manifold Extraction

We compute the manifolds of linear dependency by triangulating the regions where $\mathbf{u} \wedge \mathbf{w}_1 \wedge \cdots \wedge \mathbf{w}_k = \mathbf{0}$. In those cases, where such solution manifolds have codimension one, i.e., in the cases $k = n - 1$, they could be obtained as contours (remember that the wedge product is a scalar in such cases), using an n-dimensional variant of the marching cubes algorithm [BWC04; LC87]. On the other hand, if $k = 1$ and $n = 3$, one could use the parallel vectors extraction scheme [PR99]. However, in the remaining cases, neither of these approaches is applicable.

Instead of using a combination of these approaches, we propose to follow a generic approach that works for any n and k, inspired by the principles of marching cubes and parallel vectors extraction. We assume that the vector fields $\mathbf{u}, \mathbf{w}_1, \ldots, \mathbf{w}_k$ are given on a common n-dimensional rectilinear grid, with tensor-product multilinear interpolation. Our algorithm consists of four main steps:

1. compute solution points on (n-k)-faces of the grid (Sec. 3.2.1),
2. filter solution points (Sec. 3.2.3),
3. triangulate the solutions in each cell of the grid (Sec. 3.2.2), and
4. filter resulting k-simplices and connected components (Sec. 3.2.3).

Except for the computation of connected components, all steps are local within a cell or a cell face, and can thus be trivially parallelized. Also, only the first step needs to traverse all cells, while the subsequent steps depend on the size of the solution set. In most applications, the size of the solution set is much smaller than the number of cells.

The first step, which is the main computational bottleneck, can be sped up in some cases by rejecting those cell faces, where all of the nodes do not adhere to a certain criterion. This can always be employed in the cases of vortex core manifolds and bifurcation manifolds, where a specific number of complex eigenvalues is required. It can further be applied when extracting ridge manifolds, where the cell faces can be filtered by a scalar range in cases, where only ridges within a certain scalar range are of interest (see Section 3.3.3 for an example).

We restrict our algorithm to rectilinear grids, because its extension to arbitrary dimension is straightforward. In order to adapt the algorithm to unstructured grids, analytical derivatives of the interpolation functions within each cell, as well as the traversal of all (n-k)-dimensional cell faces would be needed.

3.2.1 Computation of Solution Points

Since the solution set $\mathcal{D}$ is k-dimensional, its intersection with the (n-k)-faces of the grid, i.e, the (n-k)-dimensional cell faces (1-dimensional faces being the cell edges), is a set of isolated points. For example, in 4D space, a 2D manifold intersects a 2D face in at most a single point (see Fig. 3.1j–3.1l). We obtain the solution points by minimizing $\|\mathbf{u} \wedge \mathbf{w}_1 \wedge \cdots \wedge \mathbf{w}_k\|$ using Gauss–Newton iteration. In a face's local coordinates $\boldsymbol{\zeta} := (\zeta_1, \ldots, \zeta_{n-k})^\top$, we choose the initial guess $\boldsymbol{\zeta}^{(0)}$ as the center of the cell face, and in each iteration, we update the local coordinates by $\boldsymbol{\zeta}^{(i+1)} = \boldsymbol{\zeta}^{(i)} - \Delta$, where

$$\Delta = \left(\mathbf{J}^\top \mathbf{J}\right)^{-1} \mathbf{J}^\top \left(\mathbf{u}(\boldsymbol{\zeta}^{(i)}) \wedge \mathbf{w}_1(\boldsymbol{\zeta}^{(i)}) \wedge \cdots \wedge \mathbf{w}_k(\boldsymbol{\zeta}^{(i)})\right), \tag{3.9}$$

and where the $l \times (n-k)$ matrix $\mathbf{J}$ is the Jacobian of Equation 3.3 in local face coordinates. By applying the product rule of differentiation to the Leibniz formula for determinants, it is obtained by differentiating each parameter of the determinant separately, and taking their sum:

$$\mathbf{J}_{ij} = \det\left(\frac{\partial \mathbf{u}}{\partial \zeta_j}, \mathbf{w}_1, \ldots, \mathbf{w}_k, \mathcal{E}_i\right) + \cdots + \det\left(\mathbf{u}, \mathbf{w}_1, \ldots, \frac{\partial \mathbf{w}_k}{\partial \zeta_j}, \mathcal{E}_i\right). \tag{3.10}$$

Since we employ multilinear interpolation within each cell face, we obtain these derivatives by analytic differentiation of the interpolation terms. These derivatives are not necessarily continuous across face boundaries, but are only used to locate solutions within the derived fields. After a maximum number of N iterations, we reject any solution, where the thresholds

$$\|\Delta\| > \vartheta_\Delta \quad \text{or} \quad \left\| \mathbf{u}(\boldsymbol{\zeta}^{(N)}) \wedge \mathbf{w}_1(\boldsymbol{\zeta}^{(N)}) \wedge \cdots \wedge \mathbf{w}_k(\boldsymbol{\zeta}^{(N)}) \right\| > \vartheta_\wedge \tag{3.11}$$

are crossed, or if $\boldsymbol{\zeta}^{(N)}$ lies outside of the cell face. The threshold ϑ_Δ serves mainly monitoring purposes, i.e., if convergence has not been achieved with respect to $\vartheta_\wedge$ but ϑ_Δ is met, increasing N might improve the result, or the other way round, if $\|\vartheta_\Delta\|$ is too large, this can indicate that the result will not sufficiently converge even with more steps. Using only one initial guess, we may obtain at most one solution point in each cell face, i.e., too high data variation within a cell face may lead to faces containing more than one solution point. We only compute one solution point per cell face, and leave more elaborate solution methods for future work.

Eigenvector Fields. If the vector fields $\mathbf{w}_1, \ldots, \mathbf{w}_k$ are eigenvectors of a tensor field, such as the Jacobian of a vector field or the Hessian of a scalar field, they need to be sorted and oriented in order to obtain smoothly varying vector fields. In order to consistently orient the eigenvector fields within a cell face $\mathcal{F}$, we perform principal component analysis (PCA) on the sets $\{\pm\mathbf{w}_j(\boldsymbol{\iota}),$ for each node $\boldsymbol{\iota}$ of $\mathcal{F}\}$, and orient the vectors $\mathbf{w}_j(\boldsymbol{\iota})$ consistent with the direction of the major component [FP01].

Enumeration of Cell Faces. To be able to traverse and store all solution points, we need a scheme for enumerating all (n-k)-faces of the n-dimensional grid. We start with an enumeration of the grid nodes, which we choose to be in scanline order. For each cell, we take its "lower left" node (i.e., the cell's node with lowest coordinate in all dimensions) as a reference. We then identify each cell's face that includes this node by the (n-k)-dimensional subspace it is embedded in, i.e., by the subset of the standard basis that spans that space. These subspaces correspond to choices of $n-k$ basis vectors, of which there are $l = \binom{n}{k}$ many. In total, the global ID of a face is thus given by the global scanline ID of the node and the type $t \in \{1, \ldots, l\}$ of the face that shares this node. Notice, how this corresponds to the definition of the wedge product, for example, the types of 2-dimensional faces of a 3D grid conceptually correspond to the wedge products $\mathbf{e}_1 \wedge \mathbf{e}_2$, $\mathbf{e}_1 \wedge \mathbf{e}_3$, and $\mathbf{e}_2 \wedge \mathbf{e}_3$. See Appendix A for details.

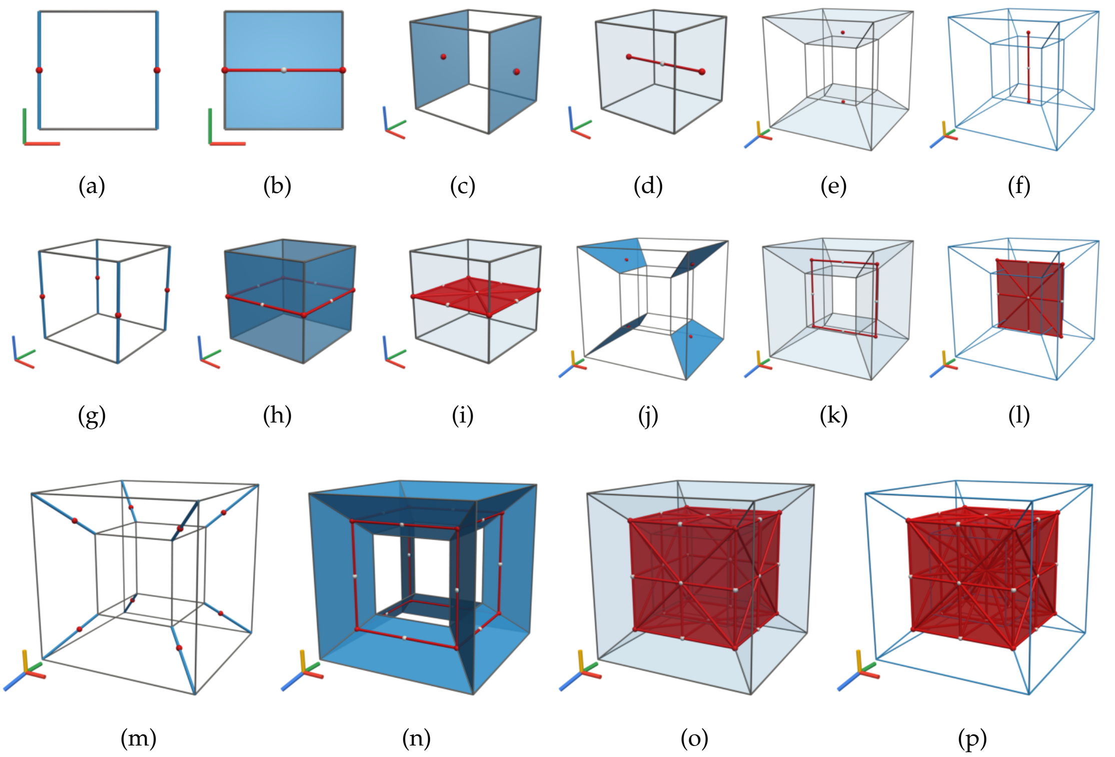


Figure 3.1: Triangulation of dependent vectors solution manifolds. Case $k = 1$ in 2D (a)–(b), 3D (c)–(d), and 4D (e)–(f) results in lines, case $k = 2$ in 3D (g)–(i) and 4D (j)–(l) results in surfaces, and case $k = 3$ in 4D (m)–(p) results in volumes. Points (red dots) on codimension k faces (blue) are connected to lines (red lines), and further to triangles and tetrahedra (in the cases $k = 2$ and $k = 3$, respectively), by inserting points at the centers of masses (white dots) of the solution points from previous steps to the triangulation. 4D cases are shown in 3D perspective projection. Currently active faces in each step of the triangulation procedure are marked in blue, with lines for 1-faces, quads for 2-faces, transparent cubes for 3-faces, and lines (wireframe) for 4-faces.

3.2.2 Triangulation

The extracted solution points are triangulated iteratively. Starting from the solution points (zero-dimensional simplices from Section 3.2.1) on the (n-k)-faces, each step connects lower-dimensional simplices that share the same cell face of one dimension higher to simplices of one dimension higher. Thus, in the i-th step, we obtain i-dimensional simplices on (n-k+i)-dimensional cell faces. For each (n-k+i)-face, the center of mass of the solution points belonging to the cell face is inserted and connected to all vertices of the (i-1)-simplices from the previous iteration, creating i-simplices. After k steps of this process, the triangulation terminates, resulting in a set of k-simplices defined within the n-dimensional cells.

Six cases of this generic triangulation approach are illustrated in Figure 3.1, which we further discuss in the following:

- $k = 1$: $n = 2$ (Fig. 3.1a,3.1b), $n = 3$ (Fig. 3.1c,3.1d), $n = 4$ (Fig. 3.1e,3.1f),
- $k = 2$: $n = 3$ (Fig. 3.1g–3.1i), $n = 4$ (Fig. 3.1j–3.1l), and
- $k = 3$: $n = 4$ (Fig. 3.1m–3.1p).

For $k = 1$, it is apparent that the intersection of $\mathcal{D}$ with a cell needs to represent points on the boundary of the cell, and that these need to be subsequently connected to a line. In 2D, the cell's boundary consists of 1-faces (Figure 3.1a). In 3D, it consists of 2-faces (Figure 3.1c), and in 4D, it consists of 3-faces (the faces of a 4D cube are 3D cubes, Figure 3.1e). Our algorithm detects those points, which we denote solution points, and connects them into lines, or in other words, into 1-simplices. We do not connect the points directly, but compute their center of mass (white points in Figure 3.1) and connect that point with each solution point, generating a 1-simplex for each solution point. This allows for more than two solution points (which are then connected to each other), and can be further applied to the construction of higher-dimensional simplices.

For $k = 2$, the intersection between $\mathcal{D}$ and a 3D cell has to represent points on 1-faces of the 3D cell (Figure 3.1g). However, we identify on the 2-faces (Figure 3.1h) of that cell the configuration from $k = 1$ in 2D (Figure 3.1a). We exploit this analogy by treating the faces of a 3D cube accordingly, i.e., connecting the points into 1-simplices (Figure 3.1h). Nevertheless, we are not done yet—to obtain a k-manifold, we connect each 1-simplex with the center of mass of all solution points, resulting in a set of 2-simplices (triangles, Figure 3.1i). In 4D, the intersection between $\mathcal{D}$ and the 4D cell are points on 2-faces of the 4D cell (Figure 3.1j). Here, we identify on the 3-faces of that cell the configuration from $k = 1$ in 3D (Figure 3.1c). We exploit this analogy by treating these faces accordingly, i.e., connecting the points into 1-simplices (Figure 3.1k). Again, we are not done yet—to obtain a k-manifold, we connect each 1-simplex with the center of mass of all solution points, resulting in a set of 2-simplices (Figure 3.1l).

For $k = 3$, the intersection between $\mathcal{D}$ and a 4D cell has to represent points on 1-faces of the 4D cell (Figure 3.1m). Here, we identify on the 3-faces of that cell the configuration from $k = 2$ in 3D (Figure 3.1g). We exploit this analogy by treating this face accordingly, i.e., connecting the points into 1-simplices (Figure 3.1n) and those to a set of 2-simplices (Figure 3.1o). And again, as it was in the case before, we are not done yet—to obtain a k-manifold, we connect each 2-simplex with the center of mass of all solution points, resulting in a set of 3-simplices (tetrahedra, Figure 3.1p).

We observe the following scheme:

- The number of steps in our algorithm depends only on k, i.e., it is independent of n. That is, for $k = 1$ we needed two steps, for $k = 2$ we needed three steps, and for $k = 3$ four steps. This increase is caused by its recursive nature, and the need for an additional step to finally establish k-manifolds.
- The extraction of the manifolds can be achieved in a recursive manner: if $k = 1$, the solutions points are obtained on the faces of the cell and connected into 1-simplices. Otherwise, an nD cube is handled by treating each of its (n-1)-faces independently as an (n-1)D cube, with a subsequent step to combine the intermediate result to the higher-dimensional simplices.

3.2.3 Filtering

Filtering takes place in two stages of our algorithm: Criteria with a pointwise definition (such as feature strength), are applied at the solution-point level. Criteria that imply connectivity, i.e., that are defined on simplex level (such as feature quality and feature size), are applied after triangulation.

As we have seen in Sections 3.1.2 and 3.1.3, applications of the DV operator typically require application-dependent filtering, such as requiring the $n - k$ respective eigenvalues of the Hessian being negative in case of ridge manifolds, or requiring the $n - k$ remaining eigenvalues of the velocity gradient to be complex in the case of vortex core manifolds. These filtering criteria are part of the feature definition, and therefore, typically do not require adjustment by the user. Nevertheless, it is common practice [PR99] to employ optional filtering by means of these criteria, denoted *feature strength*, as well as by additional measures, such as *feature quality*, and *feature size*.

Feature Strength. As mentioned in Section 3.1.2, $\mu_1, \ldots, \mu_{n-k}$ represent the second directional derivatives across the ridge manifold. In other words, the more negative they are, the more the respective profile corresponds to a peak. On the other hand, if these eigenvalues are too close to zero, ridge extraction will not be robust, as they correspond to "too flat" ridges. Thus, we provide the option of imposing a (nonnegative) threshold ϑ_r on the eigenvalues of the Hessian, i.e., we omit manifold regions where

$$- \mu_{n-k} \; \leq \; \vartheta_r \, . \tag{3.12}$$

Setting $\vartheta_r = 0$ results in the unconstrained solution (Equation 3.7).

In case of vortex core manifolds, we follow the same idea, but now filter with respect to the weakest imaginary part, since imaginary parts relate to rotational strength. Let

$\mu_1, \ldots, \mu_k$ be the real eigenvalues, we thus omit, with the (nonnegative) threshold ϑ_v, manifold regions where

$$\min_{i=k+1,\ldots,n} |\mathrm{Im}(\mu_i)| \;\leq\; \vartheta_v\,. \tag{3.13}$$

Again, setting $\vartheta_v = 0$ results in the unconstrained solution.

Bifurcation manifolds are defined by their locally separating behavior, which requires both the minor eigenvalue μ_1 and the major eigenvalue μ_n to be large at the same time. They also need to be of opposite sign, such that we omit regions, where

$$-\mu_1\mu_n \leq \vartheta_s\,, \tag{3.14}$$

and thus exhibit weak separating behavior, or are not bifurcation manifolds at all. Setting the nonnegative threshold $\vartheta_s = 0$ results in the unconstrained solution, since this only forces the signs of the eigenvalues to be alternating.

Feature Quality. Both ridge manifolds and vortex manifolds are, in case of perfect solutions, tangent to $\mathbf{u}$ [PR99]. That is, the manifolds are at the same time integral manifolds of $\mathbf{u}$, e.g., streamsurfaces in nD in case of 2-manifolds. However, local extraction methods, such as our dependent vectors operator, or the PV operator, cannot enforce such global constraints. Peikert and Roth [PR99], for the 3D case, therefore quantify feature quality by the angle between the tangent of the feature line and $\mathbf{u}$.

In our general case, we measure the angle between the n-dimensional vector $\mathbf{u}$ and the k-dimensional feature tangent space using the generalized angle between linear subspaces defined by Gunawan et al. [GNS05]. For the feature tangent $T = \langle t_1, \ldots, t_k \rangle$, the angle θ between T and $\mathbf{u}$ is given by

$$\cos^2\theta = \|\pi_T(\mathbf{u})\|^2 / \|\mathbf{u}\|^2\,, \tag{3.15}$$

i.e., the ratio of the lengths of the projection $\pi_T(\mathbf{u})$ of $\mathbf{u}$ onto T, and $\mathbf{u}$ itself. Gunawan et al. have shown that this definition reduces to the usual angles between two vectors if $k = 1$, and between a vector and a hyperplane if $k = n - 1$. For the remaining cases, this is the largest angle between $\mathbf{u}$ and all possible vectors in T.

For each vertex in the k-dimensional triangulation obtained by our algorithm, we compute its tangent space by PCA of the relative positions of the neighborhood of adjacent vertices in the triangulation. The radius of this neighborhood determines the smoothness of the tangent space across the manifold. The first k components obtained by the PCA represent an approximation of the tangent space, and are used to compute the angle between the manifold and $\mathbf{u}$.

Feature Size. Typically applied last in the filtering stage, one might want to suppress erroneous, small manifolds that are caused, e.g., by (numerical) noise. To that end, we compute the connected components of the simplicial complex using depth-first search. For each connected component, we sum over the volume of each simplex, where the volume $\mathrm{vol}(S)$ of a single k-simplex $S = \{\mathbf{v}_0, \dots, \mathbf{v}_k\}$ is given by the *Cayley-Menger determinant*. It is obtained from the squared distances of its vertices, $d_{ij} = \|\mathbf{v}_i - \mathbf{v}_j\|^2$, as

$$\mathrm{vol}(S)^2 = \frac{(-1)^{k+1}}{2^k (k!)^2} \det \begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & d_{00} & \dots & d_{0k} \\ \vdots & \vdots & & \vdots \\ 1 & d_{k0} & \dots & d_{kk} \end{bmatrix}. \tag{3.16}$$

Components, whose volume is below the nonnegative threshold ϑ_μ, are discarded. Here, $\vartheta_\mu = 0$ provides the unconstrained result.

3.3 Results

We apply the DV operator to several datasets and for several applications. First, we consider a set of synthetic vortical fields, in order to build intuition to the generalized

Table 3.1: Applied cases (k derived vector fields of dimension n) of the dependent vectors operator and extracted features of the datasets in Section 3.3. The computational costs, as well as the sizes of the respective input and output sets, are listed in Table 3.2 for the datasets (I)–(VII).

Dataset	**Grid Size**	**n**	**k**	**Feature**
1-Vortex Core (3D)	10 × 10 × 10	3	1	Vortex Core Line
1-Vortex Core (4D)	10 × 10 × 10 × 10	4	2	Vortex Core Surface
1-Vortex Core (5D) (I)	10 × 10 × 10 × 10 × 10	5	3	Vortex Core Volume
2-Vortex Core (5D)	10 × 10 × 10 × 10 × 10	5	1	2-Vortex Core Line
Double Pendulum (II)	150 × 150 × 150 × 150	4	2	Vortex Core Surface
Double Pendulum (III)	150 × 150 × 150 × 150	4	2	Bifurcation Surface
Double Gyre (IV)	1000 × 500 × 150 × 50	4	2	Valley Surface
Vortex Street (V)	41 × 241 × 41 × 16	4	2	Vortex Core Surface
Convective Flow (VI)	120 × 60 × 120 × 200	4	1	Valley Line
Convective Flow (VII)	60 × 30 × 60 × 200	4	2	Vortex Core Surface

notion of n-dimensional vortices, and second, we extract features from the 4D phase space of a double pendulum. Finally, we apply our algorithm for several features in 3D time-dependent flow simulations, where time is treated as fourth dimension, such that streamlines in these 4D space-time vector fields correspond to pathlines. All involved streamlines were computed using fourth-order Runge–Kutta integration. When applying the DV operator, all datasets use $N = 10$ as the maximum number of Gauss–Newton iterations, with tolerances $\vartheta_{\Delta} = \vartheta_{\wedge} = 10^{-11}$. The cases of the DV operator, from our experiments, are listed in Table 3.1. The complexity of the datasets, together with performance measurements, are summarized in Table 3.2.

3.3.1 Vortex Core Models in n Dimensions

We exemplify the notion of n-dimensional vortex core manifolds by considering synthetic fields, which model n-dimensional vortices based on rigid body rotations. The velocity field is given by

$$\mathbf{u}(\mathbf{x}) = \nabla\boldsymbol{\phi}(\mathbf{x})\Big(\mathbf{W}(\boldsymbol{\phi}^{-1}(\mathbf{x}))\boldsymbol{\nu}(\boldsymbol{\phi}^{-1}(\mathbf{x}))\Big), \tag{3.17}$$

where $\boldsymbol{\phi}$ transforms physical coordinates $\mathbf{x} = (x_1, \ldots, x_n)^\top$ into computational coordinates $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_n)^\top$, $\boldsymbol{\nu}(\boldsymbol{\zeta})$ describes a rigid body rotation, and $\mathbf{W}$, depending on its argument, interpolates between the identity and a matrix which transforms the rotation into a hyperbolic vector field.

Table 3.2: Details and computational cost of the datasets. Listed are the number of processed cell faces, and the number of initial solution points. Timings for computing the derivatives on (all) grid nodes (Deriv.), the dependent vectors operator (DV), filtering (F), triangulation (T), and total computation times (Σ). The timings of the triangulation step include the removal of unused vertices. See Table 3.1 for the respective grid sizes and dependent vectors dimensions.

Dataset	**Cell Faces**	**Solutions**	**Deriv.** (s)	**DV** (s)	**F** (s)	**T** (s)	**Σ** (s)
I	400 992	8761	< 1	2	< 1	< 1	2
II	2 316 092 543	444 810	433	6 293	1	19	6 746
III	255 913 981	131 082	433	4 608	1	6	5 037
II	488 933 295	6 316 775	21 703	5 105	3	52	26 863
V	8 742 615	3 064 189	5	47	1	1	54
VI	287 687 572	52 139	737	495	1	1	1 234
VII	58 646 914	385 800	12	124	< 1	< 1	136

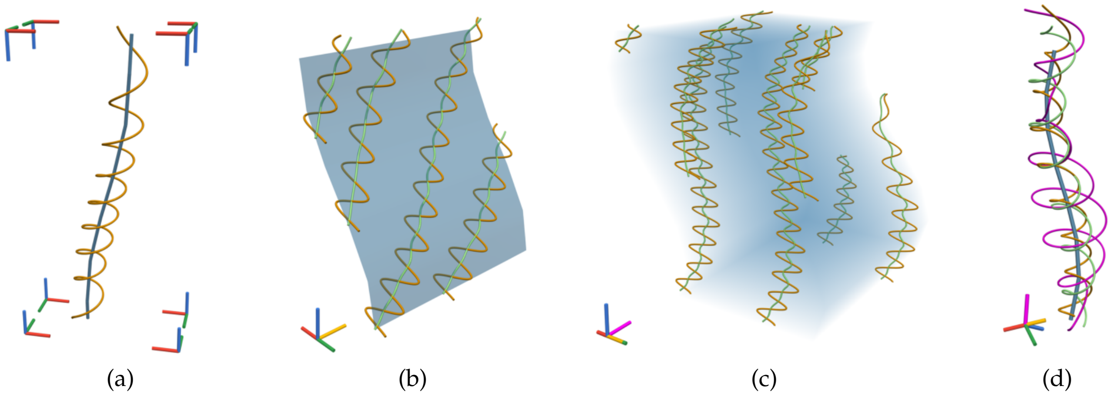


Figure 3.2: Different types of vortex core manifolds (blue) in higher dimensions. 1-vortex core manifolds, which have one plane of rotation, in 3D (a), 4D (b), and 5D (c), are one-, two-, and three-dimensional, while a 2-vortex core manifold in 5D is one-dimensional (d). The latter has two planes of rotation (orange and green streamlines), i.e., a streamline seeded at an offset of them exhibits a rotation within both planes of rotations simultaneously (magenta streamline).

The linear rigid body rotation is given by the linear field

$$\boldsymbol{v}(\boldsymbol{\zeta}) = (-\omega_1\zeta_2, \omega_1\zeta_1, \ldots, -\omega_r\zeta_{2r}, \omega_r\zeta_{2r-1}, \nu_1\zeta_{n-k}, \ldots, \nu_k\zeta_n)^\top, \tag{3.18}$$

with $2r + k = n$, consisting of r planes of rotation with respective angular velocities $\omega_1, \ldots, \omega_r$ in the first $2r$ components, and k non-rotating components with linear coefficients $\nu_1, \ldots, \nu_k$. The transformation $\mathbf{W}(\boldsymbol{\zeta})$ interpolates between the identity function, and a function that swaps the components that belong to rotations (and thus transforming the rotation to a hyperbolic field), according to a window function $w(\boldsymbol{\zeta})$, i.e.,

$$\mathbf{W}(\boldsymbol{\zeta}) = w(\boldsymbol{\zeta})\boldsymbol{\zeta} + (1 - w(\boldsymbol{\zeta}))(\zeta_2, \zeta_1, \ldots, \zeta_{2r}, \zeta_{2r-1}, \zeta_{n-k}, \ldots, \zeta_n)^\top, \tag{3.19}$$

with

$$w(\boldsymbol{\zeta}) = \sigma\left(\left|\frac{\zeta_{n-k}}{R_1}\right|^p + \cdots + \left|\frac{\zeta_n}{R_k}\right|^p - 1\right), \quad \sigma(x) = \frac{1}{1 + e^{-qx}}, \tag{3.20}$$

which, for $p, q \to \infty$, takes the value one on the k-dimensional hypercube $\mathbb{R}^{n-k} \times [-R_1, R_1] \times \cdots \times [-R_k, R_k] \subset \mathbb{R}^n$, and zero elsewhere. Choosing sufficiently large $p, q > 1$, makes the function vary smoothly, while having a sharp gradient at the boundaries of the hypercube, i.e., the resulting vector field $\mathbf{u}$ smoothly transitions between a linear saddle field and a linear rigid body rotation, such that we expect to find a k-dimensional r-vortex core manifold within the hypercube described by $w(\boldsymbol{\zeta})$.

We employ the coordinate transform $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x} + \cos(x_3)\mathbf{e}_1$ for the 1-vortex core cases and $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x} + \cos(x_5)\mathbf{e}_1$ for the 2-vortex core case. This causes the vortex core manifolds to bend along a sinusoidal path in x_1-direction. Further choosing $r = 1$, i.e., one plane of rotation, we obtain a 1-vortex core manifold, which is a line in 3D, a surface in 4D, and a volume in 5D (Figures 3.2a to 3.2c). The surface and the volume can be understood as manifolds of vortex core lines, as reflected by the 5D streamlines in the figure. In 5D space, there exist 2-vortex core lines, which exhibit two planes of rotation (Figure 3.2d). In this case, we seed two streamlines, offset in direction of each of the two complex eigenplanes, respectively (green and orange), and further a streamline that is properly contained in the 4-dimensional space spanned by the two complex eigenplanes. The resulting streamline (magenta) shows a 2-rotation. For the computation of the vortex core manifolds, we sampled all cases on 10^n regular grids, which is sufficient due to their nearly linear nature. The higher-dimensional examples are shown in a 3D orthographic projection, as indicated by the 4D and 5D orientation axes (x_1: red, x_2: green, x_3: blue, x_4: yellow, x_5: magenta).

3.3.2 Double Pendulum

We consider a double pendulum in the 2D plane, with lengths $l_1 = 10\,\mathrm{m}$, $l_2 = 1\,\mathrm{m}$ and masses $m_1 = 0.1\,\mathrm{kg}$, $m_2 = 1\,\mathrm{kg}$. As gravitational force, we choose $g = 9.81\,\mathrm{m\,s^{-2}}$. The phase space (Figure 3.3a) of this double pendulum is four-dimensional, and we describe it by the two angles θ_1, attached to l_1, θ_2 attached to l_2 (see Figure 3.3m), and their angular momenta p_{θ_1}, p_{θ_1}, i.e., the dynamical system is given by a 4D vector field $\mathbf{u}(\theta_1, \theta_2, p_{\theta_1}, p_{\theta_2}) = (\dot\theta_1, \dot\theta_2, \dot p_{\theta_1}, \dot p_{\theta_2})^\top$, with equations

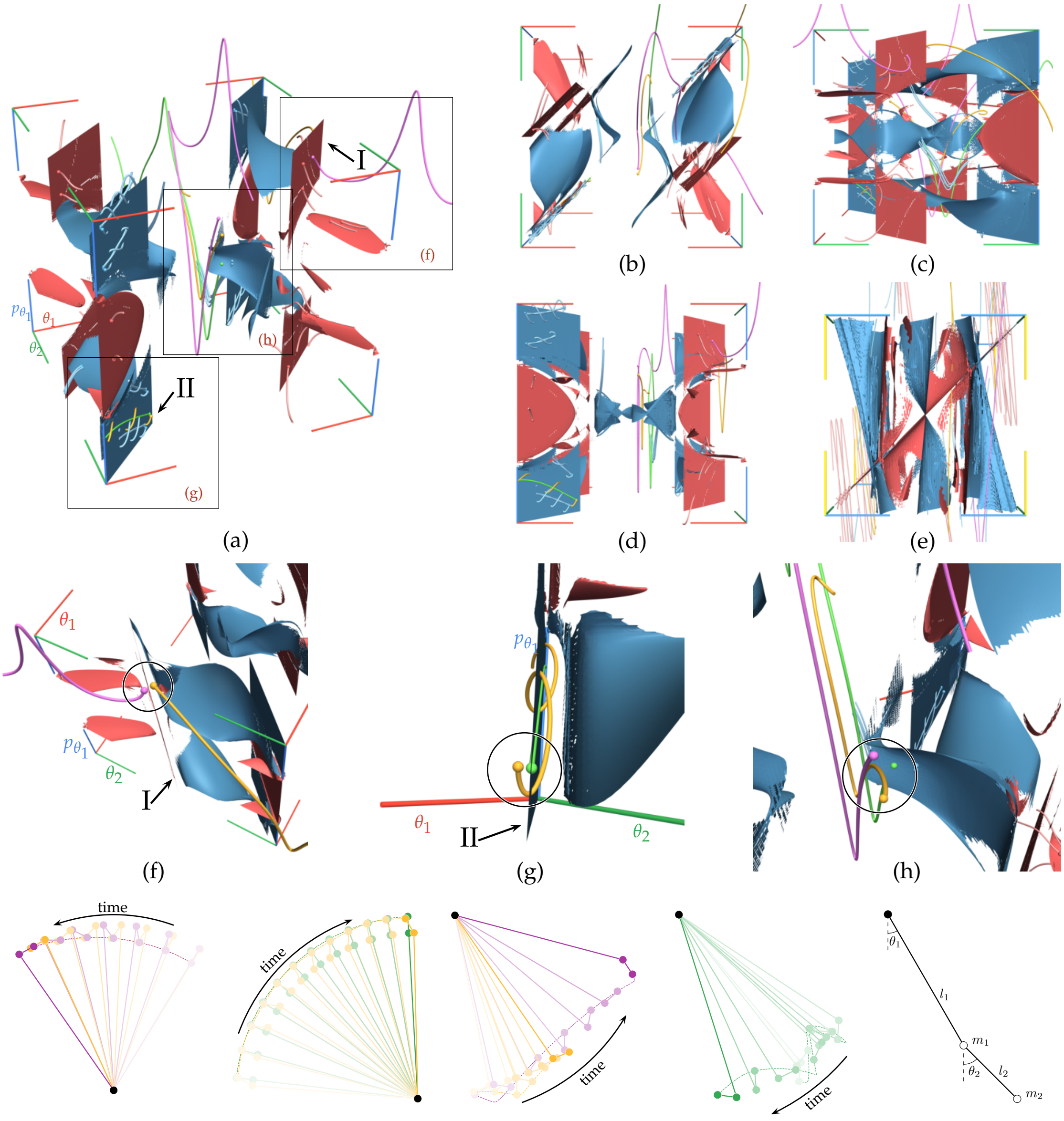
$$\dot\theta_1 = p_{\theta_1}, \quad \dot\theta_2 = p_{\theta_2}, \tag{3.21}$$

$$\begin{aligned}\dot p_{\theta_1} = {} & (m_2 l_1 p_{\theta_1}^2 \sin(\theta_2 - \theta_1)) \cos(\theta_2 - \theta_1) + m_2 g \sin(\theta_2) \cos(\theta_2 - \theta_1) \\ & + m_2 l_2 p_{\theta_2}^2 \sin(\theta_2 - \theta_1) - (m_1 + m_2) g \sin(\theta_1)) / d_1,\end{aligned} \tag{3.22}$$

$$\begin{aligned}\dot p_{\theta_2} = {} & (-m_2 l_2 p_{\theta_2}^2 \sin(\theta_2 - \theta_1) \cos(\theta_2 - \theta_1) + (m_1 + m_2) g \sin(\theta_1) \cos(\theta_2 - \theta_1) \\ & - (m_1 + m_2) l_1 p_{\theta_1}^2 \sin(\theta_2 - \theta_1) - (m_1 + m_2) g \sin(\theta_2)) / d_2,\end{aligned} \tag{3.23}$$

$$d_1 = (m_1 + m_2) l_1 - m_2 l_1 \cos^2(\theta_2 - \theta_1), \quad d_2 = (l_2 / l_1) d_1. \tag{3.24}$$

For their derivation, see, e.g., Example 6.2 of Section 6.5 in the book by Meirovitch [Mei86]. We sample the vector field on a 150^4 uniform grid on the domain $[-\pi, \pi] \times [-\pi, \pi] \times [-1, 1] \times [-1, 1]$, and extract two-dimensional vortex core manifolds and bifurcation manifolds. Since both features ideally represent manifolds



(a) (b) (c) (d) (e)

(f) (g) (h)

(i) (j) (k) (l) (m)

Figure 3.3: 4D phase space ((a)–(e)) of a double pendulum (m), with bifurcation manifolds (red), and vortex core manifolds (blue) (axes: θ_1 red, θ_2 green, p_{θ_1} blue, p_{θ_2} yellow). The trajectories of the pendulum in Euclidean coordinates are shown for selected seeds, offset in direction of the major eigenvector on a bifurcation manifold (I) (f),(i), and offset within the complex eigenplane on a vortex core manifold (II) (g),(j). Trajectories not near a bifurcation or vortex core manifold ((h),(k), magenta and yellow) exhibit behavior different than near one (f),(g). A trajectory near one of the curved vortex core manifolds ((h), green) diverges from it, while its second arm (θ_2) performs full rotations (l). The colored streamlines started from the circled seeds in (f)–(h) correspond to the motion of the pendulum in Euclidean space, shown in the same color (i)–(l). Time is indicated by saturation (older less saturated), paths of the second mass by dashed lines.

of streamlines, we filter them by discarding results that have an angle to $\mathbf{u}$ larger than 45°. While the phase space is 2π-periodic in the first two dimensions, the second two, representing angular momentum, are not periodic. Our experiments showed that extending the phase space beyond absolute value one for each momentum continuously extents the obtained manifolds. An overview of the extracted structures can be found in Figure 3.3a, where we show the 4D structures in orthographic projection onto the axes $(\theta_1, \theta_2, p_{\theta_1})$. We show bifurcation manifolds in red, and vortex core manifolds in blue. On the bifurcation surfaces we choose points, and seed streamlines offset in direction of the major eigenvector (Figure 3.3a, red lines). These streamlines show diverging behavior in vicinity of the bifurcation manifolds, thus verifying the features. We also compute streamlines offset in the complex eigenplane along vortex core manifolds, which exhibit swirling motion (Figure 3.3a, blue lines).

Furthermore, we visualize the motion of the pendulum that corresponds to a streamline in the phase space in the plane (e.g., Figure 3.3i). Two streamlines seeded on opposite sides of the bifurcation surfaces in Figure 3.3f (green and orange lines) correspond to motions of the pendulum, where the second arm tilts in opposite directions (Figure 3.3i). Similarly, we choose one seed on the vortex core manifold in Figure 3.3g (green) and one offset within the complex eigenplane (orange). The motion corresponding to the offset streamline closely follows the corresponding one on the vortex core streamline, but alternates between its opposite sides (Figure 3.3j).

Streamlines seeded neither in the vicinity of a vortex core surface nor a bifurcation surface (Figure 3.3h, magenta and yellow) exhibit none of the above behaviors (Figure 3.3k). The center of the phase space shows vortex core surfaces with high curvature, which exhibit large angles to the underlying vector field, i.e., streamlines started on them diverge. The second arm of the pendulum corresponding to a streamline started near one of them (Figure 3.3h, green) performs full rotations (Figure 3.3l).

3.3.3 Recirculation Surfaces

Recirculation surfaces are surfaces in the $(n{+}2)$-dimensional space, consisting of the spatial location $\mathbf{x}$ of dimension n together with initial time t and advection time T. They are characterized by the property that a trajectory started at $\mathbf{x}$ at time t flows back to its original position after advection time T. Recirculation surfaces have been recently proposed for flow visualization by Wilde et al. [WRT18b]. As an alternative to the authors' approach, we may extract recirculation surfaces as valley surfaces in the scalar field

$$s(\mathbf{x}, t, T) = \|\boldsymbol{\phi}_t^T(\mathbf{x}) - \mathbf{x}\| / T \,, \tag{3.25}$$

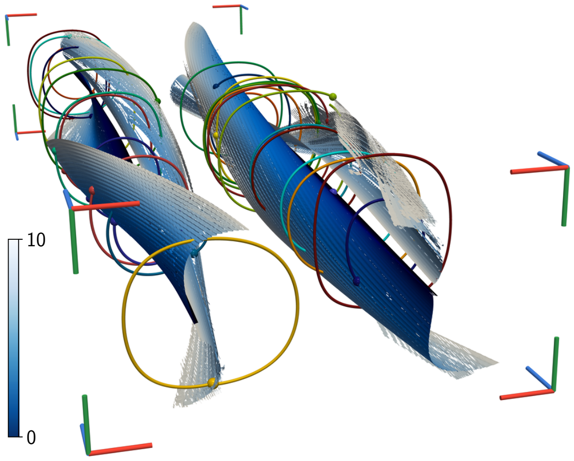


Figure 3.4: Recirculation surface of the 2D double gyre flow in space-time view (x red, y green, time blue axis). The fourth coordinate of the surface, advection time, is shown in shades of blue. Points on this surface in 4D correspond to parameters $(\mathbf{x}, t, T)$ of recirculating pathline integration. Randomly chosen points $(\mathbf{x}_i, t_i, T_i)$ on this manifold correspond to pathlines seeded at position $\mathbf{x}_i$ and time t_i, with advection time T_i, which are shown (different colors) projected onto their initial time slices t=t_i.

where $\boldsymbol{\phi}_t^T(\mathbf{x})$ denotes the flow map, which maps an initial position $\mathbf{x}$ to the endpoint of the trajectory started at time t after advection time T. While ideally, s would be exactly zero on a recirculation surface, numerically this is infeasible, since by taking the Euclidean distance s is nonnegative. Valley surfaces in the scalar field s thus allow us to approximate the true recirculation surface, where we filter out those valleys that have a scalar value above a certain threshold.

We demonstrate this approach using the synthetic 2D time-dependent Double Gyre field due to Shadden et al. [SLM05]. The scalar field s is sampled on a regular grid of size $1000 \times 500 \times 150 \times 50$ in the domain $[0,2] \times [0,1] \times [0,5] \times [0,10]$. Its computation took 187 minutes with our prototype in CUDA on a Nvidia Titan Xp. The DV operator took 7.5 hours in parallel on two 14-Core Xeon Gold 6132, where 80% of the time was spent approximating the gradients using least-squares with a 3-ring neighborhood. When computing the DV operator, we only considered those cells, where $s < 0.01$, and we further filtered the resulting solutions by rejecting those, where $s \geq 0.006$. We verified the results by computing pathlines with parameters $(\mathbf{x}, t, T)$ at randomly chosen points on the manifold. Figure 3.4 shows the projection of the solution manifold on the space-time domain $(\mathbf{x}, t)$ together with pathlines, which are projected onto the time-slices corresponding to their respective starting times.

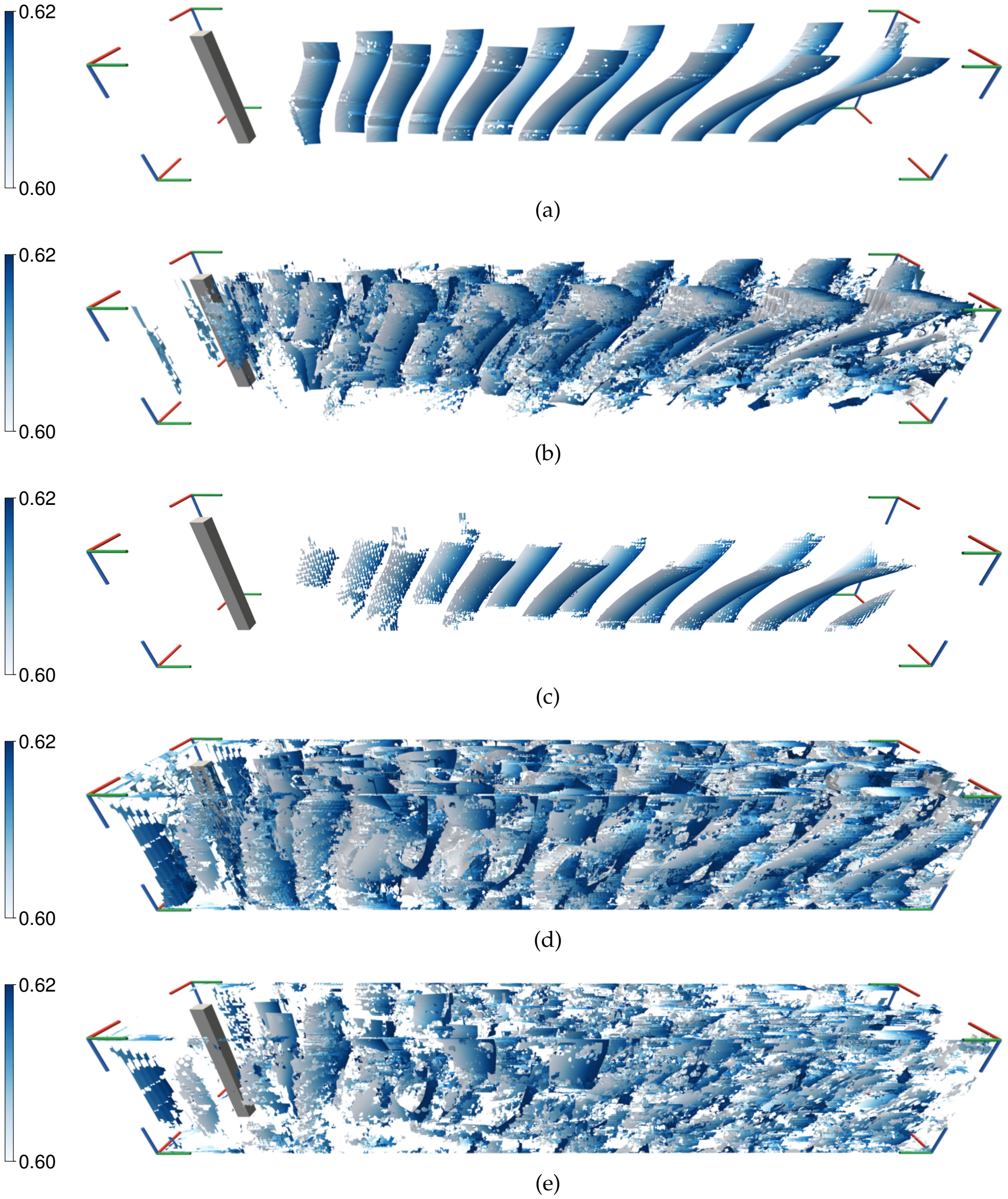
Since the computation of the valley surfaces relies on the Hessian matrix, and thus an approximation of the gradient as well as second derivatives of the flow map, our resulting surfaces exhibit holes due to aliasing in the computed flow map, especially at locations of high curvature. Wilde et al. [WRT18b] on the other hand avoid computations of gradients of the flow map, and employ local refinement of the grid, but their approach is computationally more expensive.

3.3.4 3D Von Kármán Vortex Street

Now, we demonstrate the DV operator on the time interval of $[0.6, 0.62]$ s in the spatial domain of $[0, 10] \times [0, 60] \times [0, 1]$ m^3 of a computational fluid dynamics (CFD) simulation of a von Kármán vortex street. The vortex street forms behind an obstacle and consists of vortices that move with the flow over time. The Galilean-invariant method by Weinkauf et al. [WSTH07] is able to extract these independently of the frame of reference chosen. Conceptually, the authors extract vortex core surfaces in the 4D space-time domain, which our definition (Section 3.1.3) generalizes. However, since the time component is constant one, the authors recast the 4D coplanar vectors problem (DV operator with $n = 4$, $k = 2$) as a 3D parallel vectors problem and tracked solution lines over time. We recreate this approach within our framework, by extending the two 3D vector fields $\tilde{\mathbf{u}} = \mathbf{u} + (\nabla \mathbf{u})^{-1} \mathbf{u}_t$, $\tilde{\mathbf{w}}_1 = (\nabla \mathbf{u}) \tilde{\mathbf{u}}$, where $\mathbf{u}$ denotes the time-dependent vector field from the CFD simulation, to 4D vector fields via $\mathbf{u} = (\tilde{\mathbf{u}}, 0)^\top, \mathbf{w}_1 = (\tilde{\mathbf{w}}_1, 0)^\top$ in each time slice. Adding a third, constant vector field $\mathbf{w}_2 = (0, 0, 0, 1)^\top$, the DV operator extracts surfaces, which connect the solution lines of $\tilde{\mathbf{u}} \parallel \tilde{\mathbf{w}}_1$ within each time slice to those in adjacent time slices.

We compare the approach by Weinkauf et al. (Figure 3.5a) to direct solutions in 4D with the DV operator (Figure 3.5c). The raw features of both approaches exhibit noise close to the obstacle and near the boundaries of the domain. These correspond to weakly-rotating vortices and false positives due to numerical noise, which we filtered by requiring feature strength $\vartheta_v > 10^6$. Since the DV operator not only searches for solutions in space at fixed time slices, but also in time direction, our solutions exhibit more noise (raw solutions shown in Figure 3.5b). This requires to additionally filter the solutions with angle filter of 5°, which, however, leads to false negatives near the top of the domain boundary and close to the obstacle. This is also partially caused by our simple but generic triangulation approach, which is sensitive to numerical noise. A more sophisticated triangulation algorithm could produce more accurate solutions, and less false positives when filtering.

We further consider bifurcation surfaces in this dataset. Conceptually, these are going to be an important building block in 3D time-dependent vector field topology (Section 5.3). Since, unlike vortex core manifolds, these are a much less pronounced feature, applying the DV operator yields much more noisy raw features (Figure 3.5d). In this dataset, the feature strength ϑ_s globally linearly decreases along the y-axis, making filtering by feature strength infeasible. Employing an angle filter of 5° is only able to slightly remove false positives, but also leads to many false negatives (Figure 3.5e).



(a)

(b)

(c)

(d)

(e)

Figure 3.5: Space-time domain of the 3D Von Kármán Vortex Street in the time range 0.60 s to 0.62 s. Projection into the spatial domain (x red, y green, z blue axis), with time colored in shades of blue. Swirling particle cores [WSTH07] are extracted in each time slice and tracked (a) within our framework. Vortex core surfaces extracted by the DV operator in the 4D space-time domain results in noisy raw solutions (b), which requires stronger filters (c). Bifurcation surfaces ((d), raw features) are numerically unstable due to less distinctly defined features, and thus cannot be improved by filtering by angle (e), which removes parts of the noise but also false positives.

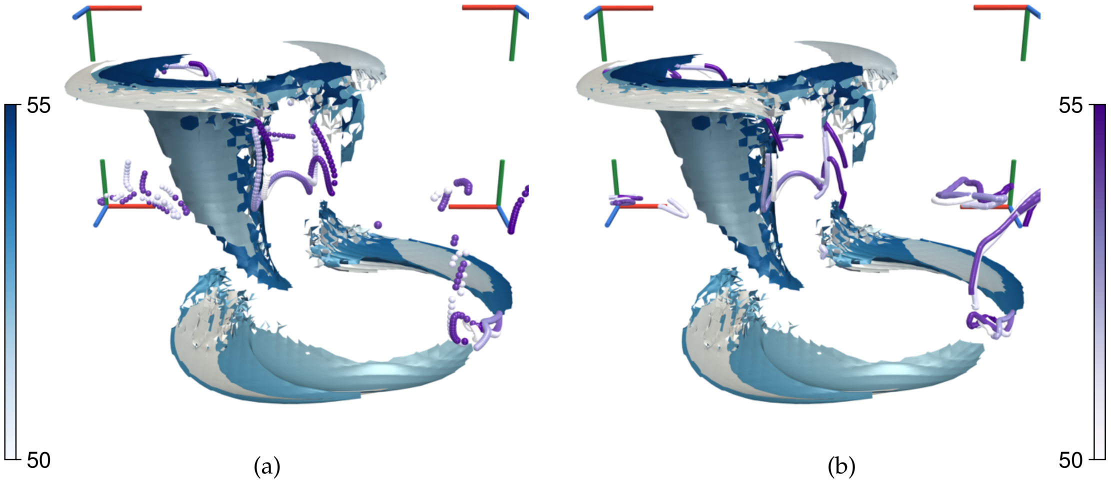


Figure 3.6: Tracking of critical points by means of valley lines of velocity magnitude in the 4D space-time domain of the 3D Convective Flow. Critical points extracted in each time step ((a), purple spheres) are tracked in the time interval $[50, 55]$s (time depicted by saturation) by valley lines ((b), purple tubes). Vortex core manifolds are further shown for context (blue surfaces).

3.3.5 3D Convective Flow

Our last example is again a time-dependent CFD simulation, but this time of a thermally driven convective flow. The dataset has a spatial resolution of $60 \times 30 \times 60$. Valley lines of the (spatial) velocity magnitude field in space-time include, among other structures, the motion of critical points of the instantaneous (spatial) field. Because the resolution of the DV operator depends on the grid, and because the original resolution caused the solution lines to be fragmented, we resampled the grid, using trilinear interpolation, in the spatial domain to a resolution of $120 \times 60 \times 120$, and consider the time interval $[50, 55]$ s, which results in a 4D grid of size $120 \times 60 \times 120 \times 200$. Since only the valleys close to zero correspond to critical points, we omit features belonging to a scalar value above 0.003. We further impose an angle threshold of 45° to account for false positives, which occur near the boundary. To put the extracted features into context, we furthermore extract vortex core surfaces (see Section 3.3.4 for a detailed discussion) in the same time interval on the original grid of size $60 \times 30 \times 60 \times 200$, where we require feature strength $\vartheta_v > 5$, and angles below 5°.

We verify the solution lines (Figure 3.6b) by extracting critical points in selected time steps by recursive subdivision of each 3D cell. The results show that extracting critical points as valley lines in the space-time magnitude field misses those that only exist for a short timespan or move farther than one cell within one time step. As the DV operator favors temporal coherence, as it treats space and time equally, critical points that are temporally unstable are missed by the extraction. Figure 3.6a shows that some critical points were not tracked by our method due to filtering false negatives and a lack of temporal coherence. As such, other methods such as (stable) feature flow fields [TS03; WTVP11] would be suited better for tracking critical points. We have included this example to demonstrate the wide applicability of our method.

3.4 Discussion

Our work focuses on developing a generic algorithm, which extracts locations of dependency of a set of k vector fields of dimension n. Recent work has also focused on generalizations of the parallel vectors operator. Oster et al. [ORT18a; ORT18b] have presented an operator, which extracts parallel vectors locations from tensor fields. The algorithm not only searches for the 3D location, but also the eigenvector itself, adding two degrees of freedom to the search space. While this is a 5D feature extraction method, it does not extract locations in a 5D vector field, where two 5D vectors are parallel (case $n = 5$, $k = 1$ of the DV operator). On the other hand, Günther and Theisel [GT18a] extracted locations in a 6D vector field, where two 6D vectors are parallel. This case corresponds to $n = 6$, $k = 1$ of the DV operator, and their formulation of a 6D cross product coincides with the notion of the wedge product used in our generalization.

While we believe that our generic algorithm enables the visualization of higher-dimensional vector and scalar fields, there are some limitations that come with its generality. Firstly, our very simple triangulation algorithm is sensitive to noise, and thus often produces non-manifold meshes. This is especially a problem for computing feature angles for filtering. An algorithm tailored specifically to, say, surfaces, could yield more accurate solutions. Secondly, the explicit computation of eigenvectors becomes less numerically stable with increasing dimension of the involved matrices, due to the iterative methods that are employed. Furthermore, the curse of dimensionality limits our approach in several ways. As dimension of the domain increases, exponentially more sample points are needed. Even though our algorithms scale linearly with the problem size, especially in 5D and beyond the computations quickly become infeasible, thus limiting us to low sampling rates. For example, the total computation time for the 10^5 dataset shown in Figure 3.2c is 2 s. With a resolution of 150^5, computation would take approximately 421 h, since computation time scales linearly with the number of cells. Secondly, memory access is a limiting factor in our implementation. In order to collect the node data of a single k-dimensional cell face, 2^k data points need to be accessed. Within a uniform grid, which is stored in scanline order, this involves accessing memory *k* times in a non-linear fashion, thus causing cache misses.

Part II

Topology

4 Visualization of 4D Vector Field Topology

In several applications, four-dimensional problems arise from lower-dimensional ones by considering additional aspects, such as time-dependency or inertia. Many of these cases, however, do not lead to true four-dimensional vector fields. Two-parameter-dependent 2D vector fields [WTHS06] only consider vectors in 2-space, while the tracking of 3D critical points [GTS04] over time only considers 3D vector fields at different instances in time. Flow-induced inertial dynamics of 2D systems [GG17], on the other hand, has an underlying 4D phase space, but the underlying flow allows the reduction of the analysis to the two-dimensional spatial domain. In this chapter, we focus on general four-dimensional vector fields, i.e., maps $\mathbf{u} : \Omega \to \mathbb{R}^4$ that assign a four-dimensional velocity $\mathbf{u}(\mathbf{x})$ to each point $\mathbf{x} \in \Omega$ in a four-dimensional domain $\Omega \subseteq \mathbb{R}^4$.

Since vector fields correspond to differential equations [Asi93] and therefore to dynamical systems, any continuous deterministic dynamical system with four-dimensional phase space, and any four-dimensional differential equation, represents a four-dimensional vector field. Besides mathematics, there is a wide range of problems that lead to such four-dimensional dynamical systems, including the change of concentration of four substances due to chemical reactions, or the evolution of four competing species. It is, however, nowadays often the case that such problems are investigated by only a few simulations instead of dense ensembles, i.e., simulations are obtained only for a few initial conditions or parameter choices. Such sets of simulations result in four-dimensional phase spaces whose domain is, however, less than four-dimensional, and thus do not necessitate four-dimensional analysis. Nevertheless, a prominent source for four-dimensional vector fields is already today the phase space in physics, consisting of position and momentum. It is $2n$-dimensional for a problem in n-space, and is used to represent the motion of inertial objects due to, e.g., forces.

Applying the concept of vector field topology to 4D vector fields, however, leads to a set of challenges, part of which we address in this chapter. First, the critical points exhibit different types in 4D fields, requiring a respective classification, and more important, appropriate visual representation. Whereas the former is a straightforward mathematical task, we address the latter with a set of glyphs, which, however, are four-dimensional. We handle the visualization of the four-dimensional structures in our approach with projection-based approaches, with a focus on avoiding projection-induced ambiguities and supporting the difficult exploration and imagination in four-dimensional space. Second, the overall navigation in 4D space, populated with the manifolds and glyphs, is of course difficult due to various paradoxes. We therefore complement our approach with a technique that helps in navigating the resulting structures, and 4D space in general. Last but not least, we present a set of vector fields that help introduce our approach, and introduce the reader to 4D space and its navigation.

Our contributions include:

- classification, glyphs, manifold extraction, and projection-based visual representation of critical points in 4D vector fields,
- and manifold-based exploration of 4D vector field topology.

4.1 Critical Points

Recall (Section 2.6.1) that a critical point $\mathbf{x}_c$ is an isolated location of vanishing velocity. In 4D vector fields, the Jacobian $\nabla\mathbf{u}(\mathbf{x}_c)$ is a real 4×4 matrix, and thus exhibits either four real eigenvalues, two real eigenvalues and a complex conjugate pair, or two complex conjugate pairs. For now, we switch to the complex plane $\mathbb{C}$ for an overview of the cases that this leads to. Due to the difficulty with 4D representation, we do not illustrate the cases in space, but exemplify them subsequently with our glyph-based approach. Table 4.1 summarizes the resulting 14 cases (which can be reduced to the eight types source, 1:3 saddle, 2:2 saddle, 1-spiral source, 1:spiral-3 saddle, 2:spiral-2 saddle, 2-spiral source, and 2-spiral saddle, if accounting for flow reversal). We keep the naming scheme from Section 2.6.1 where the numbers on each side of the colon denote the dimension of the stable (left) and unstable (right) manifold. Again, the types can be classified in terms of the number of complex conjugate pairs γ.

If there is no such pair ($\gamma = 0$), the cases are simple to classify. If all real parts have the same sign, there are the types source (Table 4.1a) and sink 4.1(b). If the real parts have both signs, we obtain saddles. If there is a single real part with opposite sign, we obtain the 1:3 4.1(c) and 3:1 4.1(e) cases, e.g., if one real part is negative (and thus, since $\det \nabla\mathbf{u}(\mathbf{x}_c) \neq 0$, three are positive), we obtain the 1:3 saddle. If there are two of each sign, we identify the 2:2 saddle 4.1(d). The 1:3 and 3:1 saddles both exhibit a 1-manifold and a 3-manifold, and since the 3-manifold is of codimension 1, it

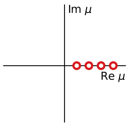
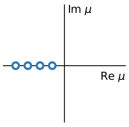
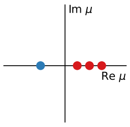
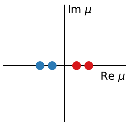
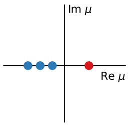
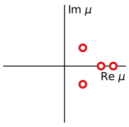
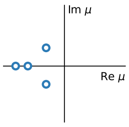
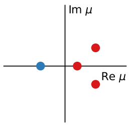
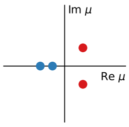
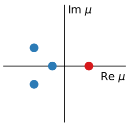
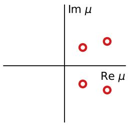
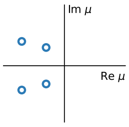
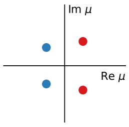
	$\rho_{1,2,3,4} > 0$	$\rho_{1,2,3,4} < 0$	$\rho_1\rho_2 < 0$	$\rho_2\rho_3 < 0$	$\rho_3\rho_4 < 0$
$\gamma = 0$	 (a) source	 (b) sink	 (c) 1:3 saddle	 (d) 2:2 saddle	 (e) 3:1 saddle
$\gamma = 1$	 (f) 1-spiral source	 (g) 1-spiral sink	 (h) 1:spiral-3 saddle	 (i) 2:spiral-2 saddle	 (j) spiral-3:1 saddle
$\gamma = 2$	 (k) 2-spiral source	 (l) 2-spiral sink	—	 (m) 2-spiral saddle	—

Table 4.1: Classification of critical point types in 4D vector fields (depicted in $\mathbb{C}$), by means of eigenvalues of $\nabla \mathbf{u}$, with real parts $\rho_{1,2,3,4}$ (ascending sorting) and number of complex pairs γ. No rotation ($\gamma = 0$), rotation in one plane ($\gamma = 1$), and rotation in two planes ($\gamma = 2$). Note that we did not depict the spiral-2:2 saddle (counterpart to (i)); it is obtained by reverting the vector field direction in (i) (mirroring about the imaginary axis). We depict repelling eigenvalues with red, attracting with blue color, separating property with a dot, and non-separating with a circle.

separates four-dimensional space. In this respect, the 1:3 and 3:1 saddles correspond to a 3D saddle (Table 2.2c). Note also that in Table 2.2c the blue arrow is not dashed, i.e., we classify the 1-manifold of 3D saddles as being separating. We do this because, although a 1-manifold does not separate 3-space, it represents an asymptote for neighboring streamlines, or in other words, it separates streamlines with respect to rotational symmetry. (Another motivation to do so is that the abovementioned section through the case in Table 2.2c, with a plane that contains the 1-manifold, results in a 2D saddle (Table 2.1c), and thus a clearly separating structure.) Consequently, we classify the 1-manifold of 1:3 and 3:1 saddles in 4D as being separating, too. This argumentation makes us classify the 2:2 saddle (Table 4.1d) as being separating as well, although 2-manifolds do not separate 4-space. As will be discussed below, we extract invariant manifolds (separatrices) from all critical point manifolds being separating in this sense.

If $\gamma = 1$ and all real parts have the same sign, we have the 1-spiral source (Table 4.1f) and the 1-spiral sink 4.1(g). We denote these cases *1-spiral* to indicate that they exhibit one plane of rotation, in contrast to *2-spiral* cases, which exhibit two planes of rotation.

(Rotations in 4-space have six degrees of freedom, and a plane of rotation rotates an object in 4D about a plane, not about a line (axis), as would be the case in 3D. Thus in 4D, rotation can take place in two linearly independent planes simultaneously.) In the remaining cases with $\gamma = 1$, the real parts have opposite sign, and thus lead to saddle behavior exhibiting rotation in one plane. If there is a single real part with opposite sign, we obtain the 1:spiral-3 4.1(h) and the spiral-3:1 4.1(j) saddles, which we abbreviate 1:s3 and s3:1 saddles, respectively, to make the distinction to 1-spiral and 2-spiral cases more clear. This notation is an extension from the 3D one, again putting the incoming manifold before the colon, and the outgoing manifold after. Thus, e.g., 1:s3 saddles 4.1(h) have inflow along a 1-manifold and outflow along a rotating (spiraling) 3-manifold. The only remaining configuration with $\gamma = 1$ is two real parts of each sign, i.e., the 2:s2 and the s2:2 saddle cases. For improved presentation in Table 4.1, we show only the 2:s2 saddle 4.1(i)—the s2:2 saddle is obtained by reversal of flow direction.

Finally, if $\gamma = 2$, there is simultaneous rotation in two independent planes, which we denote by the term "2-spiral". We have a 2-spiral source 4.1(k) if all real parts are positive, a 2-spiral sink 4.1(l) if all real parts are negative, and a 2-spiral saddle 4.1(m) in case of real parts with opposite sign.

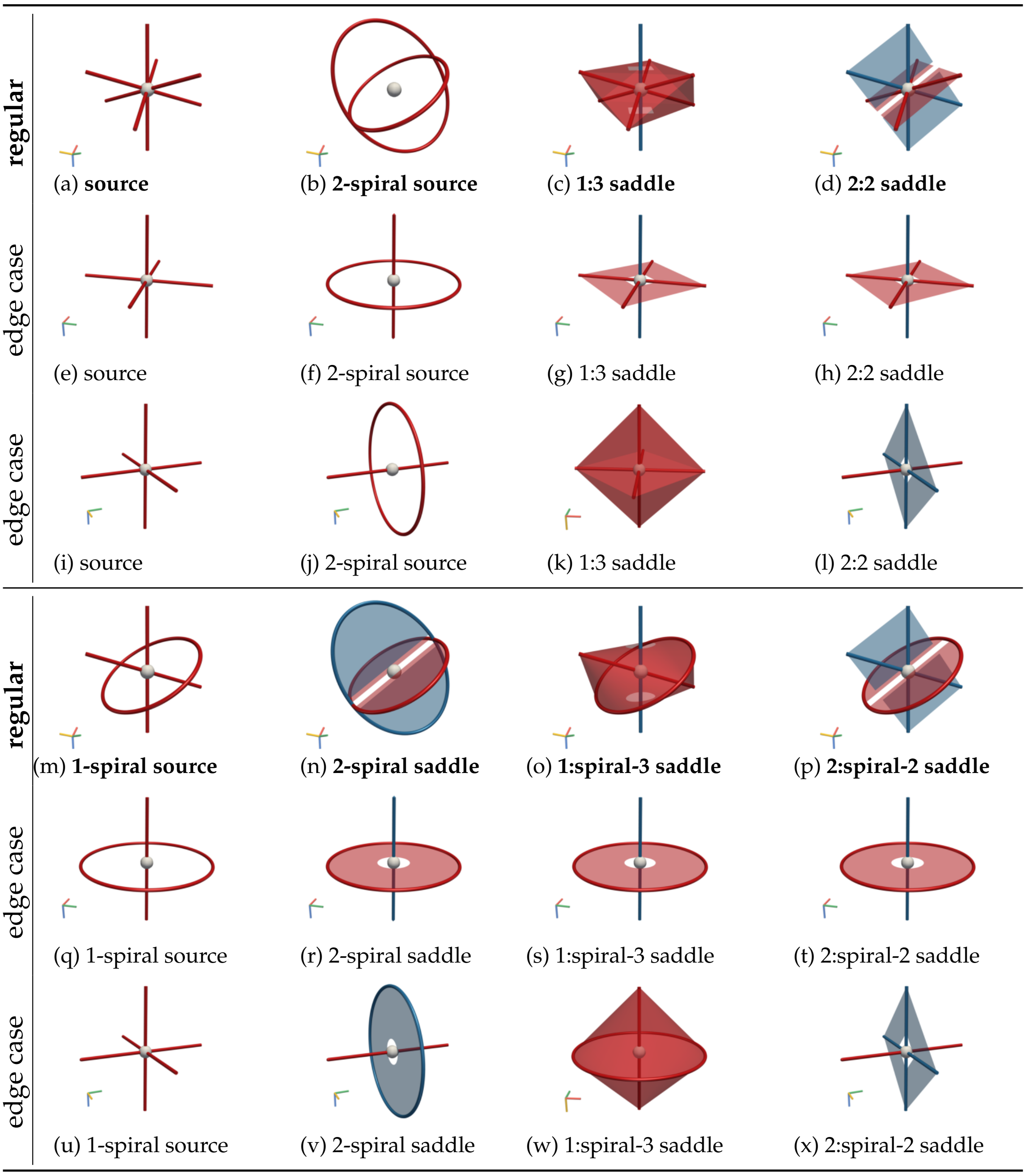
4.1.1 Extraction and Representation

Several strategies for extracting (detecting) critical points have been proposed so far. One approach is inspired by the marching cubes algorithm [LC87] for isosurface extraction. Since a critical point represents a zero in all n components of the vector field, it can be interpreted as the intersection of the n zero-level isosurfaces of its components. Technically, an early rejection test skips in these approaches all grid cells that do not exhibit a sign transition in all n components of the vector field [Wei08, Sec. 3.3.1]. In the remaining candidate cells, critical points are typically searched for by inversion of the interpolation function, or by subdivision. By their nature, these approaches work in any dimension, but the inversion approach can become challenging due to numerics and degenerate cases. In our implementation, we follow the subdivision approach, i.e., we recursively apply the candidate test and subdivide the remaining candidates until a sufficiently high resolution is achieved (or reject the refined cell if it does not span both signs in all n components). In a final clustering step, we remove duplicate candidates, which occur due to limited numerical precision mainly near cell boundaries. For subsequent classification of the determined critical points, we compute the Jacobian from the interpolation function, for consistency with the subdivision-based extraction scheme.

Having described the extraction of 4D critical points, and having classified their properties in terms of inflow, outflow, and rotation, we now can address their representation by means of glyphs. Our approach is inspired by the glyphs for 3D vector field topology by Theisel et al. [TWHS03]. Similar to the 3D approach, we build our glyphs in the respective space, i.e., our glyphs are four-dimensional. In our implementation, however, we maintain only the essential information in 4D, in particular the eigenvectors and eigenvalues of the Jacobian. The 3D geometrical representation of the glyph (tubes etc.) is built only in 3D, after projection from 4D to 3D. Generally, we represent all structures (including the invariant manifolds) in 4D space and observe them with a 4D camera. Essentially, this camera has, similar to the earlier works of Noll [Nol67] and Hollasch [Hol91], a 4D view vector and projects the 4D world to a 3D image plane. These 3D "images" can, additionally to the navigation of the 4D camera in 4D space, be navigated in 3D space. In fact, both cameras together represent the 4D camera. Further details on our projection approach are discussed in Section 4.4.

Glyphs for critical points typically consist of a set of geometric building blocks, one for each manifold type, which are then assembled according to the respective type of a critical point. Most approaches follow directly the linearized structure of the manifolds in vicinity of the critical point, i.e., the structure dictated by the eigenvalues and their eigenvectors. The skeleton of our glyph consists of the eigenvectors in 4D. We represent real eigenvectors with tubes of length proportional to the modulus of the respective eigenvalue, and map its sign to red if positive, and to blue if negative (e.g., Table 4.2a). Complex conjugate pairs are represented with circles located in the respective eigenplane, with radius proportional to the modulus of the respective real part, and again positive sign of real part mapped to red, and negative to blue (e.g., Table 4.2m).

When all real parts of the eigenvalues have same sign, the respective eigenvectors span a four-dimensional volume. Projecting this 4D volume to the 3D image plane would lead to a 3D volume, which, however, could be misleading, since there are other critical point types that exhibit a 3D manifold in 4D, which also projects in non-degenerate views to a 3D volume in 3-space. Therefore, we decided to not generate any additional geometry in these cases (Tables 4.2a, 4.2b and 4.2m, and Tables 4.1a, 4.1b, 4.1f, 4.1g, 4.1k and 4.1l). Notice that this scheme is conceptually consistent with our illustrations for 2D and 3D critical points, where we used dashed representation for non-separating cases, and solid representation for separating cases. Our 4D cases with same real part sign represent non-separating cases, and as can be see in Table 4.2, they are the only ones that consist only of lines. All remaining 4D glyphs represent separating cases (saddles), and exhibit 2-manifolds or 3-manifolds.



(a) **source** (b) **2-spiral source** (c) **1:3 saddle** (d) **2:2 saddle**

(e) source (f) 2-spiral source (g) 1:3 saddle (h) 2:2 saddle

(i) source (j) 2-spiral source (k) 1:3 saddle (l) 2:2 saddle

(m) **1-spiral source** (n) **2-spiral saddle** (o) **1:spiral-3 saddle** (p) **2:spiral-2 saddle**

(q) 1-spiral source (r) 2-spiral saddle (s) 1:spiral-3 saddle (t) 2:spiral-2 saddle

(u) 1-spiral source (v) 2-spiral saddle (w) 1:spiral-3 saddle (x) 2:spiral-2 saddle

Table 4.2: The glyphs for the eight different basic types ((a)–(d), (m)–(p)) of critical points in steady 4D vector fields, with unstable (red) and stable (blue) behavior (with flow reversal this provides glyphs for the 14 cases from Table 4.1). Each type is shown in three 4D views, leading to 3D "images". The regular view (**first and fourth row**) shows the glyph in non-degenerate projection. The second 4D view (second and fifth row) shows one edge case where manifolds degenerate due to the 3D projection, and the third 4D view (third and sixth row) shows the other edge case. Notice that in practice, these degeneracies do not occur (similar to perfectly oblique views in 3D projection), but they serve for explanatory purposes here. Observe the three-dimensional projections of the 4D axes (bottom left, x-red, y-green, z-blue, and w-orange).

We already started to depict the eigenplanes spanned by complex conjugate eigenvector pairs, i.e., planes of rotation, with *circular* discs (Tables 2.1d and 2.1e, and Tables 2.2d and 2.2e). We use this approach also in 4D, where these discs are still 2-manifold, but have a pose in 4D space (Tables 4.2n and 4.2p). On the other hand, we represent non-rotating 2-manifolds, which are spanned by real eigenvectors, with quads in 4D, whose diagonals are the two eigenvector tubes (Tables 4.2d and 4.2p). Non-rotating 3-manifolds are correspondingly represented by octahedra in 4D, whose "diagonals" are the three eigenvector tubes (Table 4.2c). The remaining case of rotating 3-manifolds, which are spanned by a real eigenvector and a complex conjugate pair of eigenvectors, can be considered a combination of a rotating 2-manifold and a non-rotating 1-manifold, leading to an intersection of two elliptic cones (Table 4.2o). Note that we represent both the octahedra and the cone intersections by their (triangulated) surface. To reduce ambiguities in projection due to degenerate projection from 4D to 3D (Table 4.2), and in particular to indicate that manifolds that intersect in their 3D projection do not intersect in 4D, we cut them out around the critical point in 4D, leading to respective gaps in the 3D projections (cf. Wang et al. [Wan+13]). Also observe that the degenerate views provided in Table 4.2 provide additional information about the critical point type itself, but are not needed when analyzing critical points in data (they serve here for complete illustration of their properties). Although such degenerate views will not appear in real-world applications with manual navigation, we make use of them for manifold-based exploration (see Section 4.5.1).

4.2 Periodic Orbits

We now classify the different types of four-dimensional periodic orbits (see Table 4.3). A Poincaré section (recall Section 2.6.2) in a 4D vector field is three-dimensional, and thus $\nabla \mathbf{P}(\mathbf{x}_c)$ has three possibly complex eigenvalues μ_1, μ_2, μ_3. The classification closely follows that of the different types of critical points in 3D vector field topology.

Periodic orbits, which have four-dimensional invariant manifolds that do not separate the domain, are shown in Tables 4.3a to 4.3f. A periodic orbit with eigenvalues $\mathrm{Im}(\mu_i) = 0$ and $0 < \mu_i < 1$ for all i is called a sink periodic orbit. If two of the eigenvalues have negative real parts $-1 < \mu_i < 0$, we call the periodic orbit a twisted sink periodic orbit. Finally, if two of the eigenvalues have non-zero imaginary parts, we call it a spiral sink periodic orbit. Note that in the case of complex eigenvalues we do not distinguish between positive and negative real parts. Replacing the condition $|\mu_i| < 1$ with $|\mu_i| > 1$, the periodic orbit becomes repelling in all perpendicular directions, and

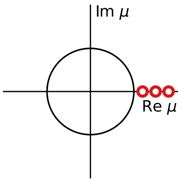
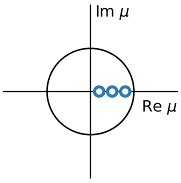
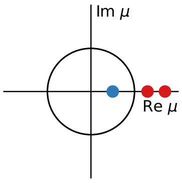
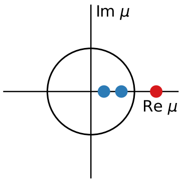
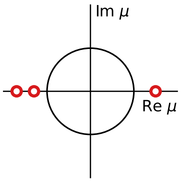
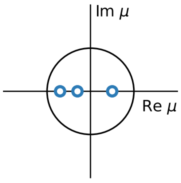
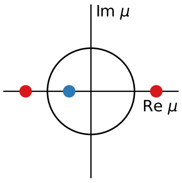
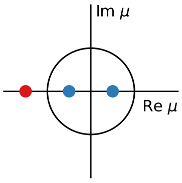
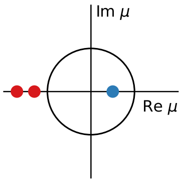
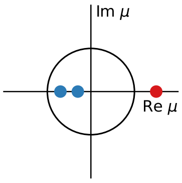
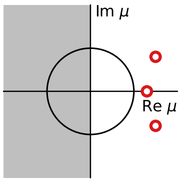
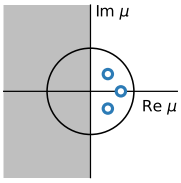
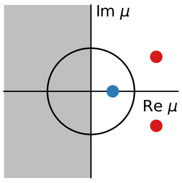
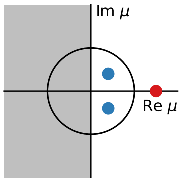
$\lvert\mu_{1,2,3}\rvert > 1$	$\lvert\mu_{1,2,3}\rvert < 1$	$\lvert\mu_1\rvert < 1, \lvert\mu_{2,3}\rvert > 1$	$\lvert\mu_{1,2}\rvert < 1, \lvert\mu_3\rvert > 1$
 (a) source	 (d) sink	 (g) 1:2 saddle	 (k) 2:1 saddle
 (b) tw. source	 (e) tw. sink	 (h) tw. 1:2 saddle	 (l) tw. 2:1 saddle
—	—	 (i) 1:2 tw. saddle	 (m) 2:1 tw. saddle
 (c) spiral source	 (f) spiral sink	 (j) 1:2 spiral saddle	 (n) 2:1 spiral saddle

Table 4.3: Classification of 4D periodic orbits by the eigenvalues of the Jacobian of the Poincaré map $\nabla \mathbf{P}$ shown in the complex plane together with the complex unit circle $z\bar{z} = 1$. The negative half-plane is grayed out in the cases, where the signs of the real parts do not matter for the classification. Repelling eigenvalues are shown in red, and attracting eigenvalues in blue color. A separating property is depicted with a dot, and non-separating property with a circle.

we obtain the three types source periodic orbit, twisted source periodic orbit, and spiral source periodic orbit.

The same scheme applies for the saddle-type periodic orbits, which we distinguish by counting the number eigenvalues with absolute value smaller and greater one. Invariant manifolds belonging to these these types of orbits are separatrices, and the eight types are shown in Tables 4.3g to 4.3n. A 1:2 saddle periodic orbit has one attracting and two repelling eigenvalues, while a 2:1 saddle periodic orbit has two attracting and one repelling eigenvalue. Since pairs of complex eigenvalues are complex conjugate, their real parts are equal. Therefore, in the case of a spiral saddle periodic orbit, only the two complex eigenvalues may have negative or positive real parts simultaneously, which we do not distinguish.

On the other hand, if all three eigenvalues are real, we distinguish between the case, where both real parts belonging to the two attracting or repelling eigenvalues are negative (1:2/2:1 twisted saddle periodic orbit), and the case, where one eigenvector of each is negative (twisted 1:2/2:1 saddle periodic orbit). In the case of a 1:2/2:1 twisted saddle periodic orbit, all streamlines contained in its 3D invariant manifold lie on a 2D twisted stream manifold. The 3D invariant manifold of a twisted 1:2/2:1 saddle periodic orbit, however, contains a 2D submanifold, which is a twisted stream manifold, and its 2D invariant manifold is twisted as well.

4.2.1 Extraction and Representation

The extraction of saddle-type periodic orbits, i.e., the topologically relevant ones, is a numerically challenging problem, because their numerical integration is typically infeasible in both forward- and backward-time direction, due to the unstable manifolds involved. This problem has not yet been sufficiently solved also in the 3D case (see Section 2.6.2), and it is even more challenging in four dimensions. Therefore, we describe a method for extraction using a Poincaré map, which we employ for illustrating the different 4D saddle-type periodic orbits in Section 4.6.6. We note that this method requires the knowledge of the location of the orbit, and relies on sufficiently accurate numerical integration. The accurate extraction of periodic orbits is left for future work.

We discretize the Poincaré map by choosing a point $\mathbf{x}_c$ in the 4D domain, and span a 3D regular grid in the three normal directions to the vector field $\mathbf{u}(\mathbf{x}_c)$ at $\mathbf{x}_c$ (Figure 4.7a, yellow sphere). At each grid node, we seed a streamline, which we integrate until the line segment between the last two consecutive integration steps intersects the 3D grid. Note that the 3D grid lies in a hyperplane in 4D space. We then store the intersection in 3D local coordinates at the grid nodes. Within this discretized field $\mathbf{P}(\mathbf{x})$, we locate fixed points by extracting critical points in the vector field $\mathbf{p}(\mathbf{x}) = \mathbf{P}(\mathbf{x}) - \mathbf{x}$. Finally, eigenvalues and eigenvectors are computed from the Jacobian $\nabla\mathbf{P}(\mathbf{x})$ at each fixed point, which we approximate by central differences.

In order to extract invariant manifolds, we seed stream surfaces along line segments spanned by the eigenvectors with length defined by the eigenvalues. By scaling by eigenvalue, we capture possible anisotropy within the stream manifolds. In the presence of 3D invariant manifolds, we further seed a closed stream manifold along the ellipse spanned by the two eigenvectors. We compute stream surfaces instead of stream volumes in the volumetric cases, since this enables us to visualize not only the 3D manifolds themselves, but also the dynamics within them.

4.3 Invariant Manifolds of Critical Points

Similar to 2D and 3D vector field topology, invariant manifolds in 4D also consist of those streamlines in $\mathbf{u}(\mathbf{x})$ that converge to saddle-type critical points in forward (stable manifolds) and reverse (unstable manifolds) direction. In the 2D and 3D case, it is common practice to build a seeding structure for the streamlines, which is based on the eigenvectors of $\nabla\mathbf{u}$. For 1-manifolds, the seeds consist of two points, each obtained by a small user-defined offset along the respective eigenvector, away from the critical point to escape its zero velocity. For 2-manifolds, one commonly places the seeds on a circle coplanar to the respective two (real parts of the) eigenvectors. Both approaches carry directly over to 4D, and we use them to obtain 1-manifolds and 2-manifolds. For 3-manifolds, finally, we put seeds on a sphere spanned by the respective three (real parts of the) eigenvectors.

Having the seeds, the 1-manifolds are obtained by simple streamline integration, resulting in a 4D polyline for each manifold. For 2-manifolds, we need to compute a streamsurface. We represent the seeding circle by a closed polyline, and employ Hultquist's streamsurface algorithm [Hul92], which can be directly employed in 4D vector fields, providing a mesh of triangles in 4-space. Finally, for the 3-manifolds, we connect the seeds into a triangulated sphere, consisting of triangles in 4-space, and employ the adaptive 3D flow volume approach by Max et al. [MBC93], which again is independent of the dimension of the surrounding space. Max et al. advance the triangular mesh with the flow and subdivide the prisms formed by two triangles at different time steps into tetrahedra. Whenever the area of a triangle reaches a threshold, they subdivide it and insert additional streamlines. Our resulting tetrahedral mesh is located in 4-space, but can be easily represented by traditional visualization toolkits, because the coordinates can be, if no 4D support is provided, represented as four scalar fields, in which case the toolkits' three coordinates can represent parametrization.

Note that, in analogy to invariant manifolds (also called separatrices) in 2D and 3D fields, they also cannot intersect in 4D, because a vector field defines at each point in space only a single direction. We store our discretized manifolds with their parametrization, i.e., with respect to streamlines and "timesurfaces". Whenever we need, for example, the normal of a 3D manifold for camera orientation, we compute it using the 4D cross product (taking three vectors) from the three vectors spanning the tetrahedron (recall that in 4D space, 3D manifolds represent hypersurfaces).

4.4 4D Camera

For a more intuitive notation that is consistent with traditional camera coordinates and 4D notations, we denote 4D space with $\mathbf{x} = (x_1, \ldots, x_4)^\top =: (x, y, z, w)^\top$ and 4D vectors, e.g., the vector field, in that space with $\mathbf{u} = (u_1, \ldots, u_4)^\top =: (u_x, u_y, u_z, u_w)^\top$. Since most display devices used for visualization are two-dimensional, we need to reduce the dimensionality from 4D to 2D. Because of its well-defined properties, its intuition, and its simplicity, we build here on projection. The traditional rendering pipeline transforms 3D world coordinates to 3D camera coordinates, and those then by projection to 2D image coordinates. The extension to our 4D setup adds one degree of freedom (DOF) for the position of the camera, and three DOFs for its orientation (we have four DOFs for position, and six DOFs for orientation in 4D). Managing these additional four DOFs is nontrivial, in particular because 4D space is difficult to interpret, and navigate. We therefore handle these degrees of freedom with two cameras: a 4D one, and a 3D one.

What we call the *4D camera* is a camera with a 4D position $\mathbf{c}$, a 4D view direction $\hat{\mathbf{c}}$ (a unit vector defining the optical axis of the camera in 4D space), and a "3D image plane", where the scene is projected to. Hence, this camera produces three-dimensional scenes. Since humans are not used to projections in 4D space, we recommend using orthographic projection within the 4D camera, leading to the setup in Figure 4.1a. The camera coordinates of this camera are $\mathbf{x}_c := (x_c, y_c, z_c, w_c)^\top =: (x, y, z, w)_c^\top$, with $(0, 0, 0, 1)_c^\top = \hat{\mathbf{c}}$. A point $(x_c, y_c, z_c, w_c)^\top$ in 4D camera coordinates is thus projected to 3D image coordinates $(x_c, y_c, z_c)^\top$. Notice that a 4D unit vector has three DOFs, so if we define the 4D camera by $\mathbf{c}$ and $\hat{\mathbf{c}}$, three DOFs remain for orientation of the camera, which fits the three degrees of orientation of a 3D camera. Thus, we do not have to accomplish the difficult definition and handling of two 4D "up vectors", as in other approaches for 4D rendering, but simply define $\mathbf{c}$ and $\hat{\mathbf{c}}$, with subsequent traditional 3D rendering *and navigation* of the "three-dimensional image" provided by the 4D camera. In other words, we treat $(x, y, z)^\top = (x_c, y_c, z_c)^\top$ for visualization of the 3D image using a traditional 3D camera. To support orientation in the original 4D space, we provide context by four additional "thumbnail 3D images" defined by $(x, y, z)^\top$, $(x, y, w)^\top$, $(x, z, w)^\top$, $(y, z, w)^\top$ that represent simple 4D projections along the w-, z-, y-, and x-axis, i.e., that are independent of the 4D camera and show its position $\mathbf{c}$ and orientation $\hat{\mathbf{c}}$ by a black sphere and an arrow, respectively. An example is shown in Figure 4.2a. Here, the top row shows the thumb-

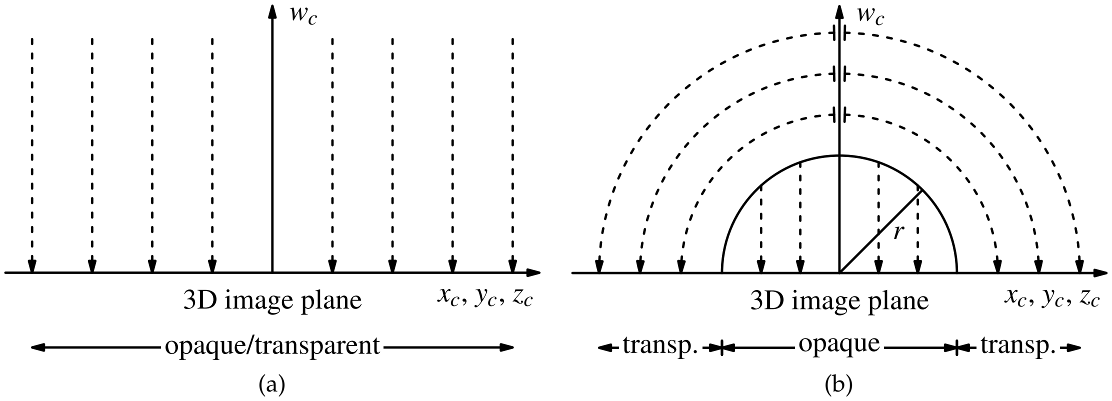


Figure 4.1: 4D clipping sphere for the removal of projection-induced intersection in 3D space. (a) Orthographic projection along w_c from 4D space into 3D image plane (depicted as line) with camera coordinates x_c, y_c, and z_c (Section 4.4). (b) Inside the sphere (solid-line circle), the same projection as in (a) is used. Outside of the clipping sphere, radial projection along the dashed curves is used instead. Both projections result in a single 3D image for interactive exploration.

nail 3D images, which the user can explore independently. Additionally, we also show the axes of the camera coordinates in the main 3D image, i.e., we show the x_c-, y_c-, z_c-, and w_c-axis in red, green, blue, and yellow, respectively (see Figure 4.2, for example). Notice that conceptually, when the 3D camera is rotated relative to the 3D image, this defines the missing DOFs of rotation of the 4D camera but does not change $\hat{\mathbf{c}}$.

4.5 Exploration Techniques

We have seen in Table 4.2 (second, third, fifth, and sixth column) that certain 4D views of the 4D camera can cause degeneracies in the resulting 3D image, leading to lower-dimensional images of manifolds. Moreover, we have seen that (i) 1-manifolds can disappear (e.g., Table 4.2q), (ii) 2-manifolds can appear as 1-manifolds (e.g., Table 4.2h), and (iii) 3-manifolds can appear as 2-manifolds (e.g., Table 4.2g). These degeneracies happen when the view direction $\hat{\mathbf{c}}$ of the 4D camera is parallel (i), coplanar (ii), or co-volumetric (iii) with the manifold at the respective point (as invariant manifolds tend to be curved in practice, this property only holds locally).

There are two main difficulties with the visualization of 3D manifolds in 4D topology: (I) visual clutter due to their volumetric appearance, and (II) (self-)intersection with manifolds due to projection; although manifolds cannot intersect in 4D space because they consist of streamlines there, their projection in 3D can intersect and tends to do so, leading to 3D images that are hard to interpret and explore. This is problematic because topological structure is often conveyed by configurations where stable and unstable manifolds meet, e.g., at critical points and saddle connectors [TWHS03].

4.5.1 Manifold-Based Exploration

Fortunately, there is a simple approach to bring 3D manifolds into oblique 4D projection, which renders them planar at least locally in the 3D image, and thus helps solve (I). Since all manifolds consist of streamlines in 4D, their 4D normal(s) is perpendicular to the streamlines, so a projection along the tangent of a streamline will bring the respective region of the manifold in degenerate 4D view, locally rendering 3D manifolds as surfaces in the 3D image, 2D manifolds as curves, and making 1D manifolds disappear. We enable the user to adjust $\hat{\mathbf{c}}$ to the vector $\mathbf{u}$ of the vector field at any time, to pick manifolds using backprojection, which moves the camera center $\mathbf{c}$ to the picked 4D point and sets $\hat{\mathbf{c}}$ to the respective precomputed streamline tangent, and also to navigate (and generate animations by moving) the camera along a selected streamline of the manifold. That way, the camera can follow a precomputed 4D streamline of a 3-manifold, keeping $\hat{\mathbf{c}}$ tangential to the streamline to show the 3-manifold locally as a plane in the 3D image. We provide a default configuration of the 3D camera by defining one "up vector" of the 4D camera to be the 4D normal vector of the manifold at $\mathbf{c}$. As the manifold is triangulated into tetrahedra, we first compute normals for each vertex of the tetrahedron that contains $\mathbf{c}$, and finally obtain the normal at $\mathbf{c}$ by barycentric interpolation. The normal of a vertex is computed as the average of the normals of all tetrahedra adjacent to it. Given a tetrahedron $(\mathbf{t}_1, \ldots, \mathbf{t}_4)$ in 4-space, its normal $\mathbf{c}$ is obtained component-wise as $\mathbf{c}_i = \det(\mathbf{t}_2 - \mathbf{t}_1, \mathbf{t}_3 - \mathbf{t}_1, \mathbf{t}_4 - \mathbf{t}_1, \mathbf{e}_i)$, $i = 1, \ldots, 4$, where $\mathbf{e}_i$ denotes the i-th standard basis vector, or equivalently, using the wedge product (Section 3.1.1), $\mathbf{c} = (\mathbf{t}_2 - \mathbf{t}_1) \wedge (\mathbf{t}_3 - \mathbf{t}_1) \wedge (\mathbf{t}_4 - \mathbf{t}_1)$. As a consequence of this choice of the 4D "up vector", the default view of the explorative 3D camera is oblique to the manifold too, i.e., its 3D "up vector" is the projection of the 4D "up vector", leading to a 2D projection that shows the manifold locally as a curve. This minimizes local occlusion and clutter, and provides a starting point for exploration by navigation of the 3D image.

We enable further exploration of the manifold, by providing the user with the option to gradually switch to neighboring streamlines of the manifold, thereby choosing the travel direction. The streamline that the camera is located on in this mode, is shown by a striped tube in the 3D image (Figure 4.6e), providing additional context and a notion of motion, which otherwise might be missing (both translation and rotation in 4D do not necessarily appear as translation and rotation in the resulting 3D image, and rotations typically involve some deformation of the 4D image, as can be seen, e.g., in Table 4.2). Of course, we enable the observer to interactively deviate $\hat{\mathbf{c}}$ from $\mathbf{u}$ to, e.g., obtain overview and context.

4.5.2 4D Clipping Sphere and Manifold Distance

To address the problem of projection-induced intersection (II) of structures in the 3D image that do not intersect in 4D space, we use two complementing approaches. On the one hand, we provide a “4D clipping sphere” that suppresses projection-induced intersection by modifying the projection from 4D to 3D. On the other hand, we visualize on each manifold the shortest 4D distance to all other manifolds, in order to convey proximity in 4D space.

Our 4D clipping sphere is centered at the camera center $\mathbf{c}$, and has an interactively defined radius r. Whereas the 4D space within the sphere is projected to the 3D image plane with orthographic projection, the outer part is projected along annular sectors to the 3D image plane (Figure 4.1b). As can be seen in this illustration, this has the effect that the outer part in 4D is also located outside the respective 3D sphere in the 3D image plane, and thus it cannot intersect with the 3D image part inside the sphere. The smaller r is chosen, the more projection-induced intersections are prevented inside the sphere, but the less regions are projected to the central part, where high-opacity rendering proved useful. The part outside the sphere is distorted in 4D, but still provides consistent context in 3D, supporting navigation. Rendering the part outside the sphere with transparency follows a “shadows on the wall” metaphor in 4D (this is shown, for example, in Figure 4.6c).

A drawback of the clipping sphere is that it constrains focus. We thus complement it by computing for each vertex of each manifold the shortest 4D distance to all vertices of all *other* manifolds.We take the logarithm of this distance field and map values close to one to high contribution in the green channel and high opacity (see, e.g., Figure 4.3f). One can see that this technique successfully reveals regions where manifolds meet in 4D; in particular it reveals saddle connectors (Section 2.6.4) in 4D. We leave the extraction of saddle connectors in 4D, however, as future work. Self-intersections of a manifold are not indicated, but these are always projection-induced because a stream manifold cannot self-intersect in 4D.

4.6 Results

In the following, we introduce datasets of increasing complexity (see Table 4.4), and use them for exemplifying our approach. The first five datasets were constructed for explanatory purposes, whereas the sixth one exemplifies the utility of our approach for analyzing the physical phase space of inertial motion. The last dataset was constructed randomly to provide a good coverage regarding complexity and variability. All of the datasets were sampled on regular four-dimensional grids consisting of quadlinearly interpolated hypercube cells. Streamlines were integrated using the fourth-order Runge–Kutta scheme (RK4) with fixed step size.

4.6.1 Linear Fields

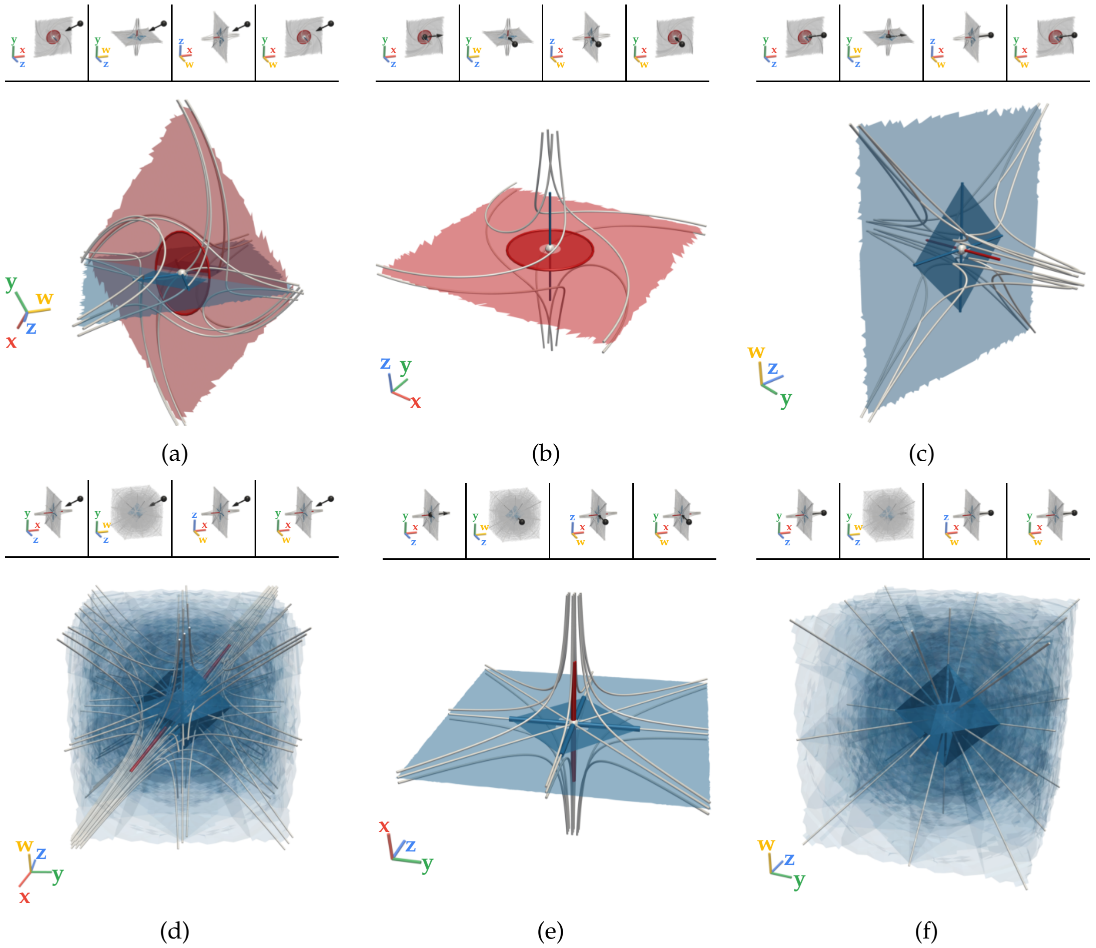
Linear fields $\mathbf{u}(\mathbf{x}) = A\mathbf{x}$ are consistent with the classification of critical points, making them an ideal entry for demonstration. A matrix that exhibits a given set of eigenvalues can be constructed using a block-diagonal matrix A with a 1×1 block for each real eigenvalue, and a 2×2 block $(c, s, -s, c)$ for each complex conjugate pair $s + ic$.

Recall that we showed the glyphs of every type of critical points using one regular projection and two degenerate ones (Table 4.2). Since our glyphs are aligned with the invariant manifolds of the linearized field, the same projections that are degenerate for the glyphs, are degenerate for the entire separatrices. In the subsequent examples, we have chosen the 4D view direction $\hat{\mathbf{c}} = (1, 1, 1, 1)^\top/2$ for depicting regular views, and the directions $\hat{\mathbf{c}} = (0, 0, 0, 1)^\top$ and $\hat{\mathbf{c}} = (1, 0, 0, 0)^\top$, resulting in degenerate views.

2:S2 Saddle We obtain a 2:s2 saddle by taking the eigenvalues $1 + i, 1 - i, -1, -1$. Such a saddle has a two-dimensional stable and a two-dimensional unstable rotating

Table 4.4: Sizes of the obtained invariant manifolds for the different datasets in Section 4.6.

Dataset	Vertices	Triangles	Tetrahedra
2:S2 Saddle	31976	61244	—
3:1 Saddle	47466	—	166548
Saddle Connector 3:1–1:3	439972	—	2000148
Saddle Connector 3:1–2:2	272979	87540	1016418
Five Critical Points	737955	538542	2101293
Acceleration Field	138016	—	449481
Random 4D Field	208961	340371	1236603



(a)

(b)

(c)

(d)

(e)

(f)

Figure 4.2: Two linear 4D vector fields with different saddle-type critical points, a 2:s2 saddle (a)–(c) and a 3:1 saddle (d)–(f). Each case is shown in a regular view (a), (d), as well as two degenerate views (b), (c), and (e), (f), which are tangential to an eigenvector of the matrix defining the vector field. While the manifolds of the 2:s2 saddle cause no occlusion in all views, the three-dimensional manifold of the 3:1 saddle causes the least occlusion in the degenerate view (e).

manifold. Both structures can be readily observed in regular views (Figure 4.2a). However, the manifolds intersect in such a view, whereas in 4D space, they only intersect in the critical point. This is revealed by the degenerate views (Figures 4.2b and 4.2c), which show that each manifold can be projected to a line that intersects the other manifold only in a point.

3:1 Saddle We obtain a 3:1 saddle by choosing the eigenvalues $-1, -1, -1, 1$. Here, the regular view (Figure 4.2d) is fully occluded by the three-dimensional stable manifold. Furthermore, we find a degenerate projection (Figure 4.2f) that fully occludes the one-dimensional unstable manifold (red) as well. Projecting tangentially to the stable manifold, i.e., in direction of one of the corresponding eigenvectors, yields the least amount of clutter (Figure 4.2e).

4.6.2 Saddle Connectors

More complex configurations are obtained by placing two linear saddle points near each other. If stable and unstable directions of the respective saddles are aligned in a certain manner, it is possible to obtain saddle connectors (Section 2.6.4). As the superposition of two linear fields is again a linear field, we introduce non-linearity by multiplication with Gaussian functions $w_i(\mathbf{x})$. Given matrices A_i, positions $\mathbf{p}_i$, and standard deviations σ_i, the vector field

$$\mathbf{u}(\mathbf{x}) = \sum_i w_i(\mathbf{x}) A_i(\mathbf{x} - \mathbf{p}_i), \quad w_i(\mathbf{x}) = \exp\left(- \sum_{j=1}^{4} \frac{(x_j - p_{ij})^2}{2\sigma_i^2} \right), \tag{4.1}$$

permits the construction of arbitrary fields with specified configurations of critical points. We leave certain parameters fixed, i.e., $\mathbf{p}_1 = (0.5, 0, 0, 0)^\top$, $\mathbf{p}_2 = (-0.5, 0, 0, 0)^\top$, and $\sigma_i = 0.5$, and vary only the matrices.

Setting $A_1 = \operatorname{diag}(-1, -1, -1, 1)$ creates a 3:1 saddle-type critical point at $\mathbf{p}_1$. We then set $A_2 = \operatorname{diag}(1, -1, -1, 1)$, or $A_2 = \operatorname{diag}(1, 1 - 1, 1)$. The former results in a saddle connector between a 3:1 saddle and a 2:2 saddle, while the latter yields a connector between a 3:1 saddle and a 1:3 saddle (Figure 4.3). Following the argumentation above, we again may find projections that show degenerate structures (Figures 4.3a, 4.3c, 4.3i and 4.3k). Again, the non-degenerate view is occluded by three-dimensional manifolds, a problem that does not exist in the degenerate views of our linear fields, where volumes appear as surfaces (Figures 4.3e and 4.3g).

None of these views, however, clearly shows the intersection of the manifolds in 4D space. Furthermore, some of the intersections in the projection are artifacts of the projection and do not exist in 4D space. Examining the distance field (Section 4.5.2), which is mapped to opacity and the green color channel, reveals proximity in 4D space (Figures 4.3b and 4.3d). Regions shown in yellow and cyan belong to an intersection (hyper-)surface between two manifolds. We see that the saddle connector between a 3:1 and a 2:2 saddle is a curve (Figure 4.3d), while the one between a 3:1 and a 1:3 saddle is a surface (Figure 4.3b). Finally, we observe that the distance field shows a spherical gradient near every critical point. This demonstrates that stable and unstable manifolds of the same critical point only intersect at a single point.

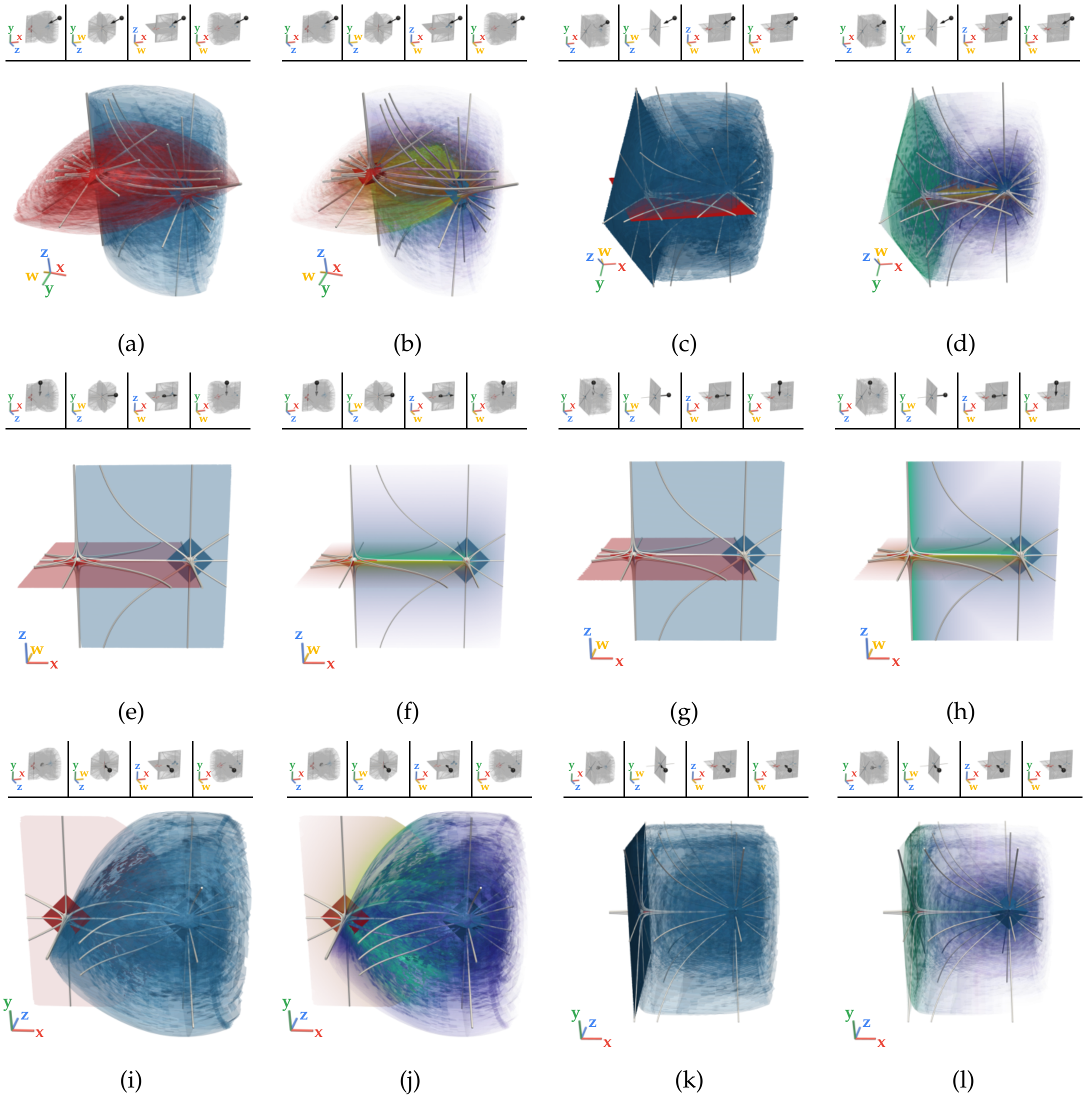


Figure 4.3: Surface-type saddle connector between a 3:1 saddle and a 1:3 saddle (first and second column), and line-type saddle connector between a 3:1 saddle and a 2:2 saddle (thrid and fourth column). Shown without (first and third column) and with (second and fourth column) distance field (green), in regular (first row), and degenerate (second and third row) views.

4.6.3 Five Critical Points

We construct a more complex example by placing five critical points using the technique described in Section 4.6.2. Here, we set all standard deviations to 0.75 and place five critical points in our vector field, a source at $(-1,-1,-1,-1)^\top$, a 1:3 saddle at $(0,0,0,0)^\top$, a sink at $(1,1,1,1)^\top$, a 3:1 saddle at $(-1,-1,1,1)^\top$, and a 2:2 saddle at $(1,1,-1,-1)^\top$. We first examine the glyphs of the critical points together with stream-

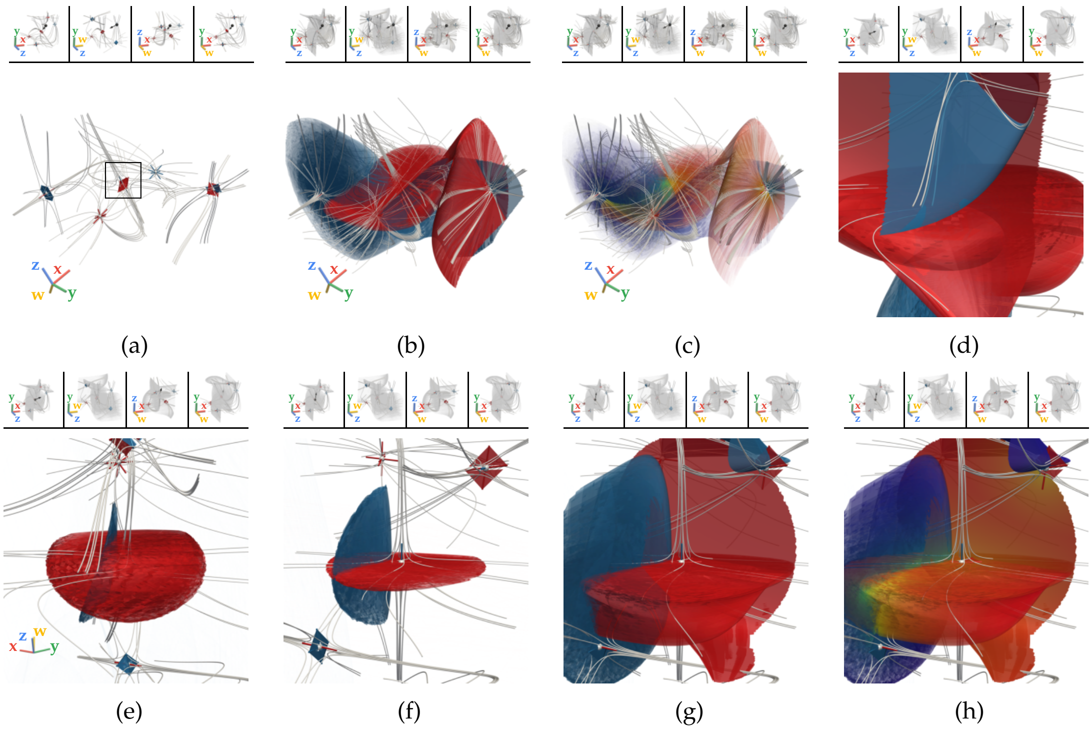


Figure 4.4: Five Critical Points dataset, overview using glyphs (a), separatrices (b), and the distance field mapped to green color (c). Initially, no clipping near the 1:3 saddle (box in (a)) is performed (d). Using a small clipping sphere (e), and aligning the view to the manifold (f) yields the least cluttered view. The saddle connectors in this dataset can be revealed using either a larger clipping sphere (g) or by the distance field (h).

lines seeded near each of the critical points (Figure 4.4a). This yields a rough overview of the vector field, with each type clearly visible. Next, we examine the same view together with all separatrices (Figure 4.4b). This projection exhibits many intersections and heavy occlusion near all critical points. The intersections in 4D space are revealed by the distance field (Figure 4.4c), which shows a saddle connector (yellow) between the 3:1 and 1:3 saddle. As opposed to the previously discussed datasets, this dataset is too complex to be examined using only the full view. As an example, we want to further examine the 1:3 saddle. Moving the 4D camera toward it leads to a cluttered view (Figure 4.4d). Using the clipping sphere, we are able to focus on the 4D neighborhood of the saddle (Figure 4.4e), however the view is still obstructed by the three-dimensional manifolds. Aligning the camera with the manifold (Figure 4.4f) lets the viewer observe an intersection of the two manifolds near the critical point. Choosing a larger radius for the clipping sphere (Figure 4.4g), allows more context to be visible, but at the same time introduces an intersection of an additional two-dimensional manifold with the three-dimensional manifold of the critical point. We may additionally use the distance field (Section 4.5.2) to distinguish intersections caused by the projection, from actual intersections in 4D space (Figure 4.4h).

4.6.4 Acceleration Field

The dynamics of inertial objects due to acceleration fields is a prominent class of problems studied in physics. There, the concept of the physical phase space is employed, consisting of position and momentum/velocity. Our example mimics the dynamics of an inertial object in a planetary system consisting of five bodies with different masses, each effecting a gravitational field, and each exhibiting an atmosphere. There are configurations where such a system of bodies has a frame of reference within which their positions do not vary, and where motion of the bodies is negligible compared to the influence of the object under consideration, also enabling a stationary model of the gravitational field. Thus, the acceleration of such an object depends solely on its position, and is defined by gravitational attraction and friction with the atmospheres.

We randomly place five point masses, i.e., the bodies, at positions $\mathbf{p}_i \in \mathbb{R}^2$ with mass $m_i \in [1,2]$ on the domain $[-5,5]^2$ (see Figure 4.5a). We assume a gravitational constant $G = 1$, and consider the force due to the gravitational field and friction ($c = 10^{-3}$) on a particle with mass m_T at position $\mathbf{p} \in \mathbb{R}^2$ and velocity $\dot{\mathbf{p}} \in \mathbb{R}^2$, which finally yields a 4D phase space vector field:

$$\mathbf{F}(\mathbf{p},\dot{\mathbf{p}}) = m_T \sum_{i=1}^{5} \left[\frac{m_i\,(\mathbf{p}_i - \mathbf{p})}{\|\mathbf{p}_i - \mathbf{p}\|^3} - \frac{c\,\|\dot{\mathbf{p}}\|\,\dot{\mathbf{p}}}{\|\mathbf{p}_i - \mathbf{p}\|^2} \right], \tag{4.2}$$

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}\begin{pmatrix} \mathbf{p} \\ \dot{\mathbf{p}} \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{p}} \\ \mathbf{F}(\mathbf{p},\dot{\mathbf{p}})/m_T \end{pmatrix}. \tag{4.3}$$

Due to the friction term, the vector field has 2-spiral sink critical points at each of the mass points and s3:1 saddle points in between them (see Figure 4.5b). Because in this special case, the x- and y-components (red and green axes) of the 4D vector field describe spatial location, while the z- and w-components (blue and yellow axes) describe velocity, we project, by default, the first two 3D thumbnail images onto the xy- and zw-planes, respectively (see Figure 4.5b top left corner). Even though this field describes a two-dimensional phenomenon, separatrices span all four dimensions (Figure 4.5c). The clutter in the projection is removed by aligning our 4D camera with one of the separatrices and using a small clipping sphere. Seeding particles on each side of a separatrix in 4D phase space demonstrates their separating property (Figure 4.5d).

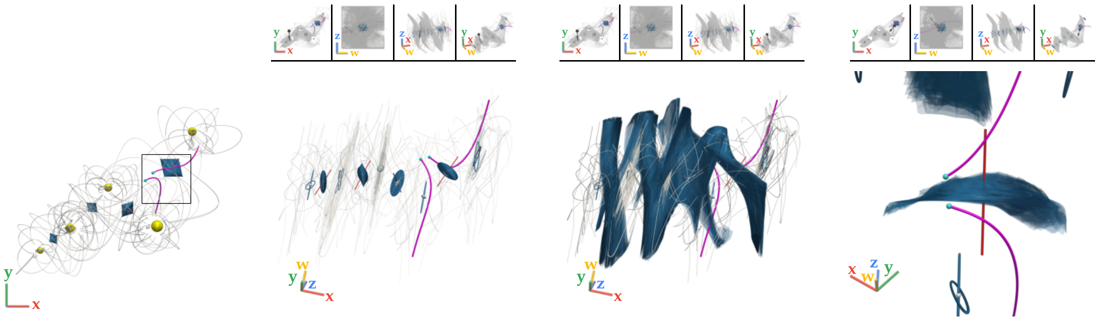


Figure 4.5: Acceleration Field dataset with five point masses (yellow spheres, scaled proportionally to mass) distributed on the 2D spatial domain (a), overview of the phase space using glyphs (b), and separatrices (c). Using a small clipping sphere, and seeding two trajectories (seeds cyan, trajectories magenta) near a saddle-type critical point (box in (a)) demonstrates the flow-separating property of the manifold in 4D phase space (d).

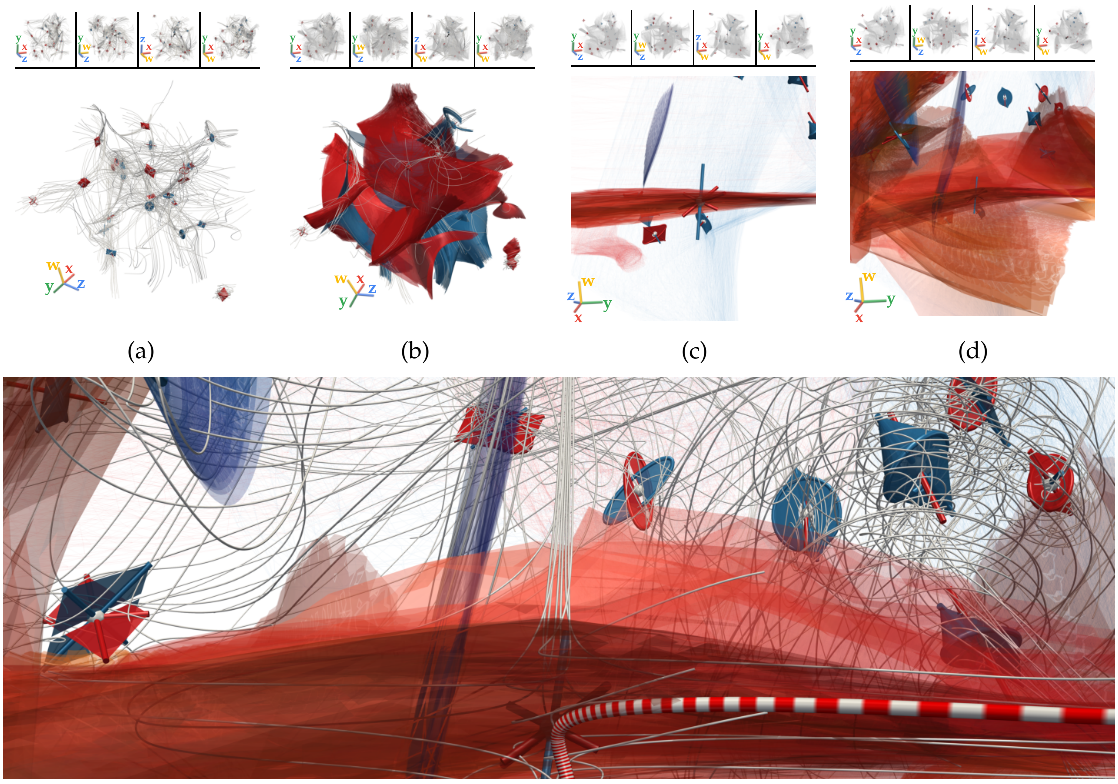
4.6.5 Random 4D Field

Finally, we generate a dataset by choosing random values on a coarse $4 \times 4 \times 4 \times 3$ grid. This dataset, shown in Figure 4.6, contains one 1-spiral sink, two 1-spiral sources, three 1:3 saddles, seven 1:s3 saddles, six s3:1 saddles, three 2:2 saddles, and one 2:s2, s2:2, and 2-spiral saddle each. An overview showing only the glyphs and streamlines seeded at the critical points shows its overall structure (Figure 4.6a). We choose a saddle-type critical point, align the camera with the corresponding manifold (Figure 4.6c), and avoid clutter in the image space, while providing context projected onto the background, using a small clipping sphere. Increasing the radius of the clipping sphere reveals more of the nearby structures (Figures 4.6d and 4.6e).

4.6.6 Periodic Orbits

To exemplify the different 4D saddle-type periodic orbits, we construct synthetic fields on the cylindrical domain $\mathbb{S}^1 \times \mathbb{R}^3$, which is embedded in $\mathbb{R}^4$ by mapping $\mathbb{S}^1$ to a circle in the (x_1, x_2)-plane, which creates a periodic orbit along this circle. For $(\alpha, \mathbf{x}_l) \in \mathbb{S}^1 \times \mathbb{R}^3$, we define a 3D vector field $\mathbf{v}(\alpha, \mathbf{x}_l) = (v_1, v_2, v_3)^\top$ using two 3D vector fields $\mathbf{u}_1(\mathbf{x}_l), \mathbf{u}_2(\mathbf{x}_l) \in \mathbb{R}^3$, where we parametrize $\mathbb{S}^1$ by $[-\pi, \pi)$:

$$\mathbf{v}(\alpha, \mathbf{x}_l) = \begin{cases} [1 - (2/\pi)|\alpha + \pi/2|] \cdot \mathbf{u}_1(\mathbf{x}_l), & \text{if } \alpha < 0, \\ [1 - (2/\pi)|\alpha - \pi/2|] \cdot \mathbf{u}_2(\mathbf{x}_l), & \text{if } \alpha \geq 0. \end{cases} \tag{4.4}$$



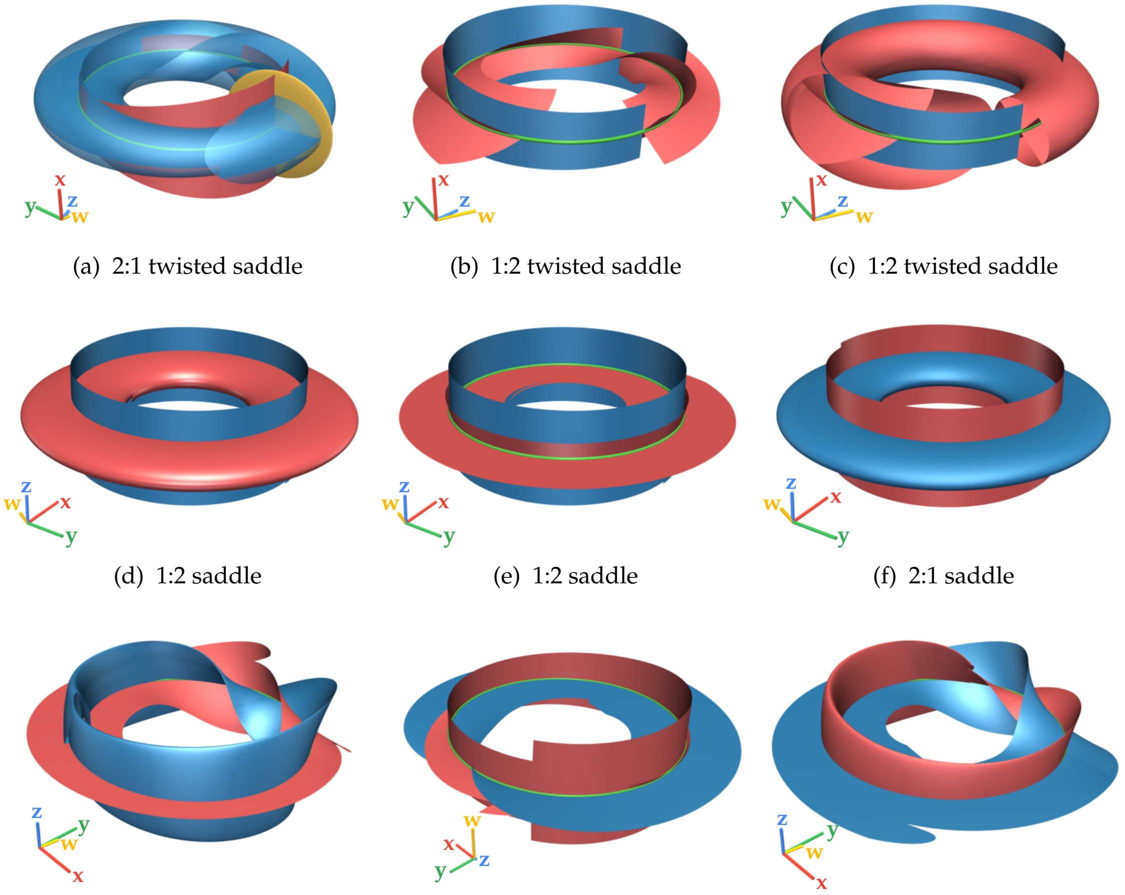
(a) (b) (c) (d)

(e)

Figure 4.6: Random 4D Field dataset, with global structure shown using glyphs (a), and separatrices (b). The dataset is explored by aligning the view to a manifold (c) near a saddle point, providing a planar 3D projection, with further context projected in the background. Increasing the radius of the clipping sphere reveals more structures that are nearby (d). The last view with streamlines is shown in (e), where the 4D camera moves in 4D space along a selected streamline (striped red–white), while maintaining a view which minimizes the clutter in the 3D projection.

The 4D vector field is then $\mathbf{u} = (\cos(\alpha)v_1 - \omega x_2, \sin(\alpha)v_1 + \omega x_1, v_2, v_3)^\top$, where $\omega = 2\pi/\tau$ for a chosen orbital period τ. In the orthogonal profile along the periodic orbit, the vector field linearly transitions between $\mathbf{u}_1$, $\mathbf{0}$, and $\mathbf{u}_2$.

Defining $\mathbf{u}_1 = (-x_1, -x_2, x_3)^\top$, $\mathbf{u}_2 = (\tau/2) \cdot (-x_2, x_1, 0)^\top$ yields a 2:1 twisted periodic orbit (Figure 4.7a), while reversing the signs of $\mathbf{u}_1$ yields a 1:2 twisted periodic orbit (Figure 4.7c). In these cases, any streamsurface within the 3D invariant manifold performs a 180° twist along the orbit, which is controlled by the factor $\tau/2$ (smaller factors would result in spiral saddle-type periodic orbits). We show this behavior by additionally seeding stream surfaces along the two eigenvectors that span the seeding circle of the 3D invariant manifold (Figure 4.7b).



(a) 2:1 twisted saddle (b) 1:2 twisted saddle (c) 1:2 twisted saddle

(d) 1:2 saddle (e) 1:2 saddle (f) 2:1 saddle

(g) twisted 1:2 saddle (h) twisted 1:2 saddle (i) twisted 2:1 saddle

Figure 4.7: The different types of 4D saddle periodic orbits (green lines), with their invariant manifolds (blue: stable, red: unstable) seeded from the Poincaré section (yellow sphere). The center column shows the 3D invariant manifolds without their circularly seeded surface.

Using $\mathbf{u}_1 = \mathbf{u}_2 = (-x_1, -x_2, x_3)^\top$ results in a 2:1 saddle periodic orbit and similarly in a 1:2 saddle periodic orbit by reversing signs (Figures 4.7d and 4.7f). In this case, stream surfaces within the 3D manifolds are not twisted (Figure 4.7e).

Finally, we define $\mathbf{u}_1 = (-x_1, -x_2, x_3)^\top$, $\mathbf{u}_2 = (\tau/2) \cdot (-x_3, 0, x_1)^\top$, which results in a twisted 2:1 periodic orbit and a twisted 1:2 periodic orbit with reversed signs. The projections in Figures 4.7g and 4.7i are aligned with the two eigenvectors, whose eigenvalues have negative real parts, and thus make the 3D manifolds appear to be twisted around the 2D manifolds. In Figure 4.7h, we only show the stream surfaces seeded along the eigenvectors, in a projection, which is tangential to the 2D rotation plane that defines the twisting behavior. This view reveals that in 4D space we have an orthogonal direction, in which the 3D manifolds do not exhibit twisting behavior. This behavior is only possible due to the additional degree of freedom in four-dimensional space, and does not exist in the case of 3D periodic orbits.

4.7 Discussion

Our approach focuses on critical points and their invariant manifolds. While these are able to provide qualitative insight into a 4D vector field, for a more complete description, several building blocks are still missing. We already presented a classification of 4D periodic orbits, but are missing a generic extraction algorithm. This is a not yet entirely solved problem, also in the case of 3D saddle-type periodic orbits [KRRS14; WS02]. Further topological structres include saddle connectors [TWHS03], boundary switch curves [WTHS04], invariant tori, and bifurcation lines [MBES16; MSE13].

There are some special cases, where 4D vector fields are obtained from spatially lower-dimensional phenomena. The first is regarding the space-time domain of a 3D time-dependent vector field as a 4D steady vector field. Such fields do not contain any critical points, since time moves constantly in forward direction (i.e., the last component is constant 1), and thus different approaches need to be employed, such as tracking of critical points [GTS04] or time-dependent topology, which is subject of the next chapter. Second, the physical phase space of 2D steady inertial dynamics is a four-dimensional vector field, where two components correspond to space and two components to momentum. While, as presented by Günther and Gross [GG17], dynamics of flow-induced inertial dynamics can be reduced to the spatial dimension, general inertial dynamics need to be considered in the full $2n$-dimensional phase space. Further, being restricted to the n-dimensional spatial domain, Günther and Gross cannot show separatrices in the $2n$-dimensional phase space.

The case of time-dependent inertial dynamics adds another dimension, and is covered by Sagristà et al. [Sag+17]. Our work can be applied to 2D steady inertial dynamics; however, we currently do not distinguish the different units of position and momentum. For a quantitative analysis, a more specific adaption would be needed. We show such an example in Section 4.6.4. Since our method relies on separatrices, and thus focuses on the qualitative analysis of the system, mixing scales and units is not an issue.

Unfortunately, we were unable to find complex (simulated) real-world datasets. Even though many problems with, e.g., a three-dimensional spatial domain depend on a fourth dimension such as pressure or concentration, most numerical simulations treat these as attribute only, instead of computing dense ensembles. Such simulations thus do not output the full four-dimensional state space, but a three-dimensional submanifold for, say, isolated initial conditions. We hope to see such data in the future.

5 Time-Dependent Vector Field Topology

Time-dependent vector fields defined on 2D or 3D spatial domains, describe the motion of massless particles over time. They arise, for example, from computational fluid dynamics, where numerical simulations yield the motion of fluids over a finite time interval. Treating time as additional dimension, their associated steady space-time vector fields are three- or four-dimensional. As a special case of our dependent vectors operator, feature extraction in 4D space-time vector fields was discussed in Chapter 3.

Since the last component of the space-time flow is constant one, steady 3D or 4D vector field topology (see Section 2.6 and Chapter 4) cannot be applied directly, i.e., it would yield an empty structure. Applying VFT to each time step of the dataset, and thus freezing time at each instance, on the other hand yields structures that can be tracked over time to give insights into the data [TWHS04a]. However, these (Eulerian) structures have no direct meaning for the time-dependent (Lagrangian) motion of particles. They are especially not invariant under changes of frames of references, and even a well-chosen frame of reference cannot capture time-dependent flow structure [HH89].

Instead, the notion of Lagrangian coherent structures is commonly used for analyzing the flow of time-dependent vector fields. They are defined as those material lines or surfaces [SLM05] that are locally the most attracting or repelling transport barriers, and are included in the set of ridges of the finite-time Lyapunov exponent fields [Hal00]. It has been shown [SW10] that ridges in the forward- and backward-time FTLE Fields intersect in distinguished hyperbolic trajectories (DHTs). These are pathlines that exhibit attracting and repelling manifolds in the space-time phase space, which coincide with the LCS. In this sense, DHTs take on the role of saddle-type critical points from steady VFT, i.e., they represent points, which move over time, and generalized streaklines take on the role of their separatrices. In 3D flows, FTLE ridge intersection are typically lines rather than points [ÜSE13], i.e., the ridges intersect in hyperbolic path surfaces (HPSs) instead. These take on the role of bifurcation lines, which we include in steady VFT.

The main challenge posed by time-dependent vector field topology is the extraction of the DHTs and HPSs. In order to obtain intersections of ridges in forward and reverse FTLE fields, a dense grid of particles would need to be integrated in forward and backward time. To ensure robust extraction of the ridges, this would need to be done at very high resolution (or in an adaptive manner). However, performing this costly computation at a single instance of time would not be enough: numerical integration started at such an intersection point exponentially accumulates errors along the repelling or attracting manifolds in forward or backward time, and additionally DHTs can appear and disappear over time. So far, two local extraction methods have been proposed to solve this problem. Machado et al. [MBES16] locally track 2D critical points in a Galilean-invariant reference frame, and locally refine the solution toward the closest pathline. Secondly, Baeza Rojo and Günther [BG20] solve a least-squares optimization problem to obtain more accurate frames of reference, in which critical points follow pathlines more closely. While the local refinement scheme of Machado et al. does not need to converge to a DHT or at all, paths of critical points in an optimal frame of reference, as computed by Baeza Rojo and Günther, do not need to be DHTs or even pathlines, as we are going to show in Section 5.6.1.

Both approaches have in common that they are mostly local in nature. In this chapter, we bridge the gap between local and global integration-based techniques. First, we locally extract candidate lines from tracked critical points and bifurcation lines in appropriately chosen reference frames. Second, we globally compute the dynamics in the localized flow along each initial candidate, using numerical integration. This is similar to computing the localized FTLE [Kas+09] along a single pathline, but with the path given by the initial candidate instead of a pathline. Finally, closely following the works of Ide et al. [ISW02] as well as Branicki and Wiggins [BW09], we compute a time-dependent coordinate transformation, which decouples repelling and attracting directions into one-dimensional systems, which allows for separate integration along the repelling direction in backward time and along the attracting direction in forward time, thus refining the initial candidate toward the DHT. Having obtained a set of DHTs and HPSs, we compute streak manifolds seeded along them, which represent LCS, and thus the topology of time-dependent flow over its entire time interval.

This chapter is organized as follows. In Section 5.1, we introduce the notion of distinguished hyperbolic trajectories due to Ide et al. [ISW02], its use for refinement of initial candidates, and our extension to hyperbolic path surfaces. Based on this, we propose methods for the extraction of hyperbolic trajectories and hyperbolic paths surfaces in Sections 5.2 and 5.3. In the remainder of the chapter, we discuss results of our method in the 2D and 3D cases. This discussion is organized into properties of our proposed time-dependent topology (Section 5.4), analysis of convergence and performance of our method (Section 5.5), and finally evaluation on numerical datasets (Section 5.6). This chapter is concluded in Section 5.7.

Our contributions include:

- We combine existing local DHT extraction techniques with global numerical approaches [BW09; ISW02] and refinement.
- We obtain consistent offsets for seeding streak manifolds and propose an automatic selection of seeding lengths.
- We evaluate different approaches to obtain DHT candidates.
- We compare our technique with previous approaches, and validate our results using the FTLE.
- We show that no purely local technique is able to accurately extract LCS-based features in general time-dependent flow.

Specifically for the 3D case:

- We introduce a local extraction of HPS candidate surfaces in 4D space-time.
- We present a robust and accurate refinement of these candidates to HPSs.
- We demonstrate the necessity to additionally include unsteady equivalents of bifurcation lines, spiral saddle critical points, a class of saddle-type periodic orbits, and saddle connectors.

5.1 Distinguished Hyperbolic Trajectories

We consider a pathline $\mathbf{x}(t)$ of a 2D or 3D flow over the time interval $[t_0, t_N]$, together with the fundamental solution matrix $\mathbf{X}_{t_0}(t)$ of its localized flow (Equation 2.26), and the singular value decomposition $\mathbf{X}_{t_0}(t) = \mathbf{B}_{t_0}(t)e^{\mathbf{\Sigma}_{t_0}(t)}\mathbf{R}_{t_0}(t)^\top$ (Equation 2.27). The discussion in this section closely follows the works of Ide et al. [ISW02] and Ju et al. [JSW03].

If the pathline $\mathbf{x}(t)$ is structurally stable, its localized flow can be transformed into the linear system

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{D}_{t_0}(t_N)\mathbf{y}(t), \tag{5.1}$$

which is defined by the diagonal, time-independent matrix

$$\mathbf{D}_{t_0}(t_N) = \frac{1}{t_N - t_0}\mathbf{\Sigma}_{t_0}(t_N) = \operatorname{diag}\Big(\lambda^1_{t_0}(t_N), \lambda^2_{t_0}(t_N)\,[, \lambda^3_{t_0}(t_N)]\Big), \tag{5.2}$$

where $\lambda^i_{t_0}(t_N)$ denote the finite-time Lyapunov exponents (Equation 2.28). This is achieved by the time-dependent coordinate transformation $\mathbf{y}(t) = \mathbf{T}(t)\boldsymbol{\delta}_{\mathbf{x}}(t)$ [ISW02], defined over the time interval $[t_0, t_N]$,

$$\mathbf{T}(t) = e^{(t-t_0)\mathbf{D}_{t_0}(t_N)}\mathbf{R}_{t_0}(t_N)^\top \mathbf{R}_{t_0}(t) e^{-\boldsymbol{\Sigma}_{t_0}(t)}\mathbf{B}_{t_0}(t)^\top . \tag{5.3}$$

In spatial coordinates, the perturbation directions belonging to the Lyapunov exponents $\lambda^i_{t_0}(t)$, i.e., the Lyapunov vectors, are obtained as $\boldsymbol{\xi}^i_{t_0}(t) = \mathbf{T}^{-1}(t)\mathbf{e}_i$, where $\mathbf{e}_i$ denote the vectors of the standard basis, or, up to a scalar factor as $\boldsymbol{\xi}^i_{t_0}(t) = \mathbf{X}(t)\mathbf{R}_{t_0}(t_N)\mathbf{e}_i$. These perturbation directions are represented by the components in Equation 5.1.

A structurally stable pathline is characterized by none of the diagonal entries of $\mathbf{D}_{t_0}(t_N)$ being zero. It is attracting if all entries of $\mathbf{D}_{t_0}(t_N)$ are negative, and repelling if all entries are positive. A hyperbolic trajectory has both positive and negative entries in $\mathbf{D}_{t_0}(t_N)$ in this notation. Since with this notion, almost all trajectories are hyperbolic, Ide et al. [ISW02] define a DHT as a hyperbolic trajectory, which remains in a bounded neighborhood $\mathcal{B}$ for all time, while all other trajectories starting in $\mathcal{B}$ leave $\mathcal{B}$ at exponential rate in forward and backward time, and, in addition, the trajectory is not an intersection of attracting and repelling manifolds of other DHTs. The latter condition is important (Section 5.4.5), since such non-distinguished trajectories manifest themselves as false-positive ridge intersections of the forward and backward FTLE fields, but they are not topological generators of the time-dependent flow. This definition of a DHT is more general than the notion of Haller [Hal00], since it does not require instantaneous hyperbolicity $\det \mathbf{u} < 0$. On the other hand, it can be shown [Cop78, p.50ff] that under similar preconditions instantaneous hyperbolicity implies the existence of a DHT. For a discussion of the reverse, we refer the reader to Branicki and Wiggins [BW10, p.16f].

We now use the preceding discussion to obtain a refinement scheme, which corrects an initial candidate line $\tilde{\mathbf{x}}(t)$ toward a nearby DHT. Using the same definitions as above, we change into the coordinate frame $\mathbf{w}(t) = \mathbf{T}(t)(\mathbf{x}(t) - \tilde{\mathbf{x}}(t))$. The flow of the trajectory $\mathbf{x}(t)$ then takes the form

$$\frac{d}{dt}\mathbf{w}(t) = \mathbf{D}_{t_0}(t_N)\mathbf{w}(t) + \mathbf{h}(\mathbf{w}(t), t), \tag{5.4}$$

where the nonlinear part $\mathbf{h}(\mathbf{w}(t), t)$ is given by [ISW02]

$$\begin{aligned}\mathbf{h}(\mathbf{w}(t), t) =& \mathbf{T}(t)\,\mathbf{u}\Big(\mathbf{T}^{-1}(t)\mathbf{w}(t) + \tilde{\mathbf{x}}(t), t\Big) \\ & - \mathbf{T}(t)\nabla\mathbf{u}(\tilde{\mathbf{x}}(t), t)\mathbf{T}^{-1}(t)\mathbf{w}(t) - \mathbf{T}(t)\dot{\tilde{\mathbf{x}}}(t).\end{aligned} \tag{5.5}$$

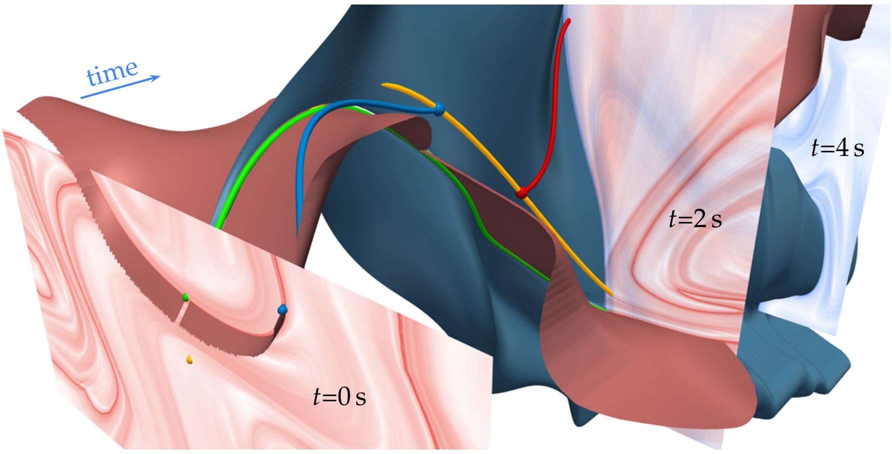


Figure 5.1: 2D pathline integration, shown in space-time view, started at a point on an initial path $\tilde{\mathbf{x}}(t)$ (yellow) near a DHT (green) in forward time (red line) is attracted by the attracting LCS (blue surface), and repelled by the repelling LCS (red surface). The roles are reversed for backward integration (blue line), thus making DHT extraction by naive numerical integration infeasible. Decoupling these two integration directions along the initial path is used as basis for refinement toward the DHT. At the time boundaries of the hyperbolic region, only one of the directions (repelling at lower time boundary, attracting at upper time boundary) can be refined, because no attracting or no repelling behavior (see FTLE, no saddle-type behavior) are present.

In Equation 5.4, attracting and repelling behavior is decoupled into one-dimensional systems. Thus, we are able to integrate along attracting directions in forward time, and along repelling directions in backward time. An *approximate DHT* for a given finite-time interval $[t_0, t_N]$ is the solution of the integral equations [ISW02]

$$\mathbf{w}_i(t) = \begin{cases} \int_{t_0}^{t} e^{d_i(t-s)}\mathbf{h}_i(\mathbf{w}(s), s)\mathrm{d}s, & \text{if } d_i < 0, \\ -\int_{t}^{t_N} e^{d_i(t-s)}\mathbf{h}_i(\mathbf{w}(s), s)\mathrm{d}s, & \text{otherwise,} \end{cases} \tag{5.6}$$

where $\mathbf{w}_i$ and $\mathbf{h}_i$ denote the ith component of $\mathbf{w}$ and $\mathbf{h}$, and d_i the diagonal entries of $\mathbf{D}_{t_0}(t_N)$. Since integration starts and stops at the time boundaries of the initial candidate, a solution $\mathbf{w}(t)$ takes the special values

$$\begin{aligned} \mathbf{w}_i(t_0) &= 0, \quad \text{if } d_i < 0, \\ \mathbf{w}_i(t_N) &= 0, \quad \text{otherwise.} \end{aligned} \tag{5.7}$$

The decoupled integration in Equation 5.6 reverses the repelling behavior of the DHT and thus corrects toward it in both forward and backward time (see Figure 5.1). This also explains the initial values in Equation 5.7: the DHT does not exist across these time boundaries, and integration beyond them, even if the data would permit it, would not undergo the same hyperbolic behavior and thus not yield a correction. Finally, we obtain a refinement of the initial candidate $\tilde{\mathbf{x}}(t)$ toward an approximate DHT,

$$\tilde{\mathbf{x}}_{\mathrm{DHT}}(t) = \tilde{\mathbf{x}}(t) + \mathbf{T}(t)^{-1}\mathbf{w}(t). \tag{5.8}$$

5.1.1 Distinguishedness in the Finite-Time Setting

In the case of infinite integration time, i.e., in the limit of $t_0 \to -\infty$ and $t_N \to \infty$ in Equation 5.6, the approximate DHT $\tilde{\mathbf{x}}_{\mathrm{DHT}}$ converges to the actual infinite-time DHT [ISW02], given that the initial candidate is within a bounded region, depending on its Lyapunov exponents [JSW03]. In typical fluid flows, however, hyperbolic regions and thus the hyperbolic trajectories are bounded regardless of the length of the time interval [Hal00]. As further noted by Haller [Hal15], the notion of a DHT in a finite-time setting would identify all but a countably infinite set of trajectories.

Indeed, a finite-time distinguished hyperbolic trajectory [BW10, Def. A.6 ff.] needs to be defined as a trajectory $\mathbf{x}(t)$, which can be decomposed into a sum of an initial candidate and the correction toward an approximate DHT (Equation 5.8). This definition does not yield unique, isolated trajectories. Instead, the choice of an initial candidate selects a particular subset of the possible finite-time DHTs. The particular choice employed in our method is evaluated in Section 5.6.

Note that a mathematical rigorous definition of a bifurcation line in steady vector fields, as introduced by Machado et al. [MSE13], would be analogous. Bifurcation lines, as extracted by the authors, are those streamline segments that can be obtained from initial candidates using their proposed refinement (see Section 2.5.5).

5.1.2 Hyperbolic Path Surfaces

Existing literature [BW09; ISW02] only considers isolated, distinguished HTs, typically obtained from paths of critical points. However, as we discuss next, in the 3D case it is necessary to consider *surfaces* consisting of (non-distinguished) HTs.

Intersections of hyperbolic ridges in the forward- and reverse-time FTLE fields include hyperbolic trajectories. They are those trajectories that stay in an instantaneously hyperbolic region locally for the longest time [Hal01], and can also be regarded as a finite-time approximation of distinguished hyperbolic trajectories (DHTs) [ISW02]. Note that in 3D flows, such ridge surface intersections are in general curves (Figure 5.2), and thus represent surfaces in the space-time domain [ÜSE13], not curves. This means that due to the finite time interval trajectories that pass through these intersection curves

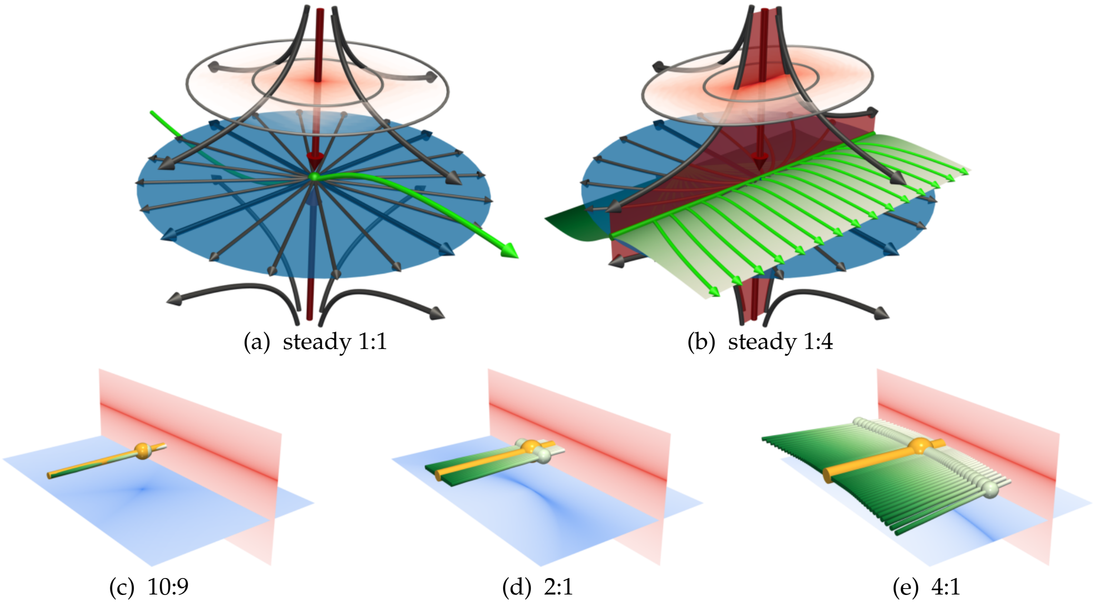


Figure 5.2: Hyperbolic trajectories (HT) for a 3D saddle-type critical point (with different anisotropy ratios of $\lambda_1 : \lambda_2$) moving from left to right. FTLE ridges (red/blue) at a steady saddle intersect in critical point (a) in the isotropic case, and in bifurcation line in the anisotropic case (b). In time-dependent flows, only almost isotropic (c) configurations exhibit isolated HTs, whereas a single HT (orange) only captures a subset of the the FTLE ridge intersection in generic anisotropic cases (d),(e). Hyperbolic path surface with time from green to white, forward (red) and backward (blue) FTLE slices shown at the time depicted by the spheres, where FTLE ridges intersect in a point ((c), ridge line and ridge surface), and in a line ((d),(e), two ridge surfaces).

cannot be distinguished and all equally contribute to the organization of the flow. We call these surfaces hyperbolic path surfaces (HPS), if they do not represent intersections of our streak manifolds generated from hyperbolic path surfaces, i.e., they are generators (see Section 5.4.5) of the unsteady topology.

5.2 Extraction of Hyperbolic Trajectories

We extract the time-dependent vector field topology, defined by hyperbolic trajectories, of a vector field $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^n$ given on a time interval $[t_0, t_N]$ using the following steps:

1. Locally extract a set of initial candidate lines $\tilde{\mathbf{x}}(t)$.
2. Refine the initial candidates toward DHTs.
3. Determine seeding lengths at each DHT.
4. Seed streaklines along the obtained DHTs.

In the three-dimensional case n=3, we additionally extract hyperbolic path surfaces, which we discuss in Section 5.3.

5.2.1 Candidate Extraction

Candidates $\tilde{\mathbf{x}}(t)$ for DHTs are obtained by tracking saddle-type critical points in a suitable frame of reference. Given a decomposition

$$\mathbf{u}(\mathbf{x},t) = \mathbf{w}(\mathbf{x},t) + \mathbf{f}(\mathbf{x},t), \tag{5.9}$$

into observer motion $\mathbf{f}(\mathbf{x},t)$ and an observed vector field $\mathbf{w}(\mathbf{x},t)$, we obtain initial polylines as paths of critical points in $\mathbf{w}$, which we filter by instantaneous hyperbolicity $\mathrm{Re}(\mu_1)\mathrm{Re}(\mu_n) < -\tau_h$ with a positive threshold τ_h. Here, μ_i denote the eigenvalues of $\nabla\mathbf{w}$ sorted by their real parts, i.e., hyperbolicity is the product of the largest and smallest real parts. In the 2D case, hyperbolicity is equal to $\det\nabla\mathbf{w}$.

This extraction can be formulated using the dependent vectors operator in the (n+1)-dimensional space-time domain as $\bar{\mathbf{w}}(\mathbf{x},t) \wedge \bar{\mathbf{e}} = \mathbf{0}$, where $\bar{\mathbf{w}}(\mathbf{x},t) = (\mathbf{w}(\mathbf{x},t),0)^\top$ and $\bar{\mathbf{e}} = (\mathbf{0},1)^\top$. Similarly to the approach of Machado et al. [MBES16] in the 2D case, this dependent vectors formulation corresponds to tracking critical points in $\mathbf{w}$ by connecting those critical points that move at most one cell size of the spatial grid between two time steps. This requires the dataset to have an adequate temporal resolution.

Conceptually, the 2D and 3D cases are the same. However, our experiments show that not all approaches work equally well in both cases, as we outline below.

Extraction in 2D

For 2D flows, we choose the threshold τ_h rather large in order to obtain short but robust solution lines. These are then extended by seeding at their endpoints streamlines in the observer motion field $\mathbf{f}$ in forward and reverse direction, until integration leaves the hyperbolic region ($\det\nabla\mathbf{u} \geq 0$), or a domain boundary is reached (e.g., Figure 5.23f). Notice that $\mathbf{f}$ is not a feature flow field of $\mathbf{w}$ in general, since it neglects possible motion of observed critical points in $\mathbf{w}$ relative to the observer motion $\mathbf{f}$. In Sections 5.6.1 and 5.6.2, we evaluate using candidates in the lab frame [ISW02], the Galilean-invariant frame of reference defined by the feature flow field [MBES16], and displacement-invariant frame of reference [BG20]. Since it consistently resulted in the best results in our experiments, and comes with moderate additional computational costs, we propose to employ the displacement-invariant frame of reference [BG20] for typical datasets.

Extraction in 3D

For 3D time-dependent vector fields, we employ the Galilean-invariant frame of reference defined by the feature flow field $\mathbf{f} = -\nabla\mathbf{u}^{-1}\mathbf{u}_t$. In order to avoid inversion of the matrix $\nabla\mathbf{u}$, we instead solve the linear least squares problem

$$\int_{\mathbf{x}\in\mathcal{U}} \|\nabla\mathbf{u}(\mathbf{x},t)\mathbf{f}(\mathbf{x},t) + \mathbf{u}_t(\mathbf{x},t)\|^2 \rightarrow \min \tag{5.10}$$

for the unknown $\mathbf{f}(\mathbf{x},t)$ at each time step over a local neighborhood $\mathcal{U}$ of 10 grid nodes, with $\mathbf{u}_t(\mathbf{x},t) := \partial\mathbf{u}(\mathbf{x},t)/\partial t$, using the method by Günther and Theisel [GT20] (see Appendix B.1 for details). Generally, any reference frame that minimizes a time derivative [BG20; GT20], could be employed. However, in our experiments, we were unable to achieve more consistent results than using a Galilean-invariant frame of reference (e.g., missing features in Figures 5.31a and 5.32b). We also found integration along observer motion to be unstable in the 3D case (Figures 5.30a and 5.30b). This has, however, no impact on the Galilean invariance or objectivity of the obtained topology (Section 5.4).

5.2.2 Refinement

The initial line $\tilde{\mathbf{x}}(t)$ is given as a polyline $(\mathbf{x}_0, t_0), \dots, (\mathbf{x}_N, t_N)$ in space-time. The Jacobian matrix is computed at each of these space-time locations, $\mathbf{J}_0 = \nabla\mathbf{u}(\mathbf{x}_0, t_0), \dots, \mathbf{J}_N = \nabla\mathbf{u}(\mathbf{x}_N, t_N)$. Using linear interpolation in time, the singular value decomposition (Equation 2.27) of the fundamental solution matrix $\mathbf{X}(t, t_0)$ is computed by numerical integration of the initial value problem

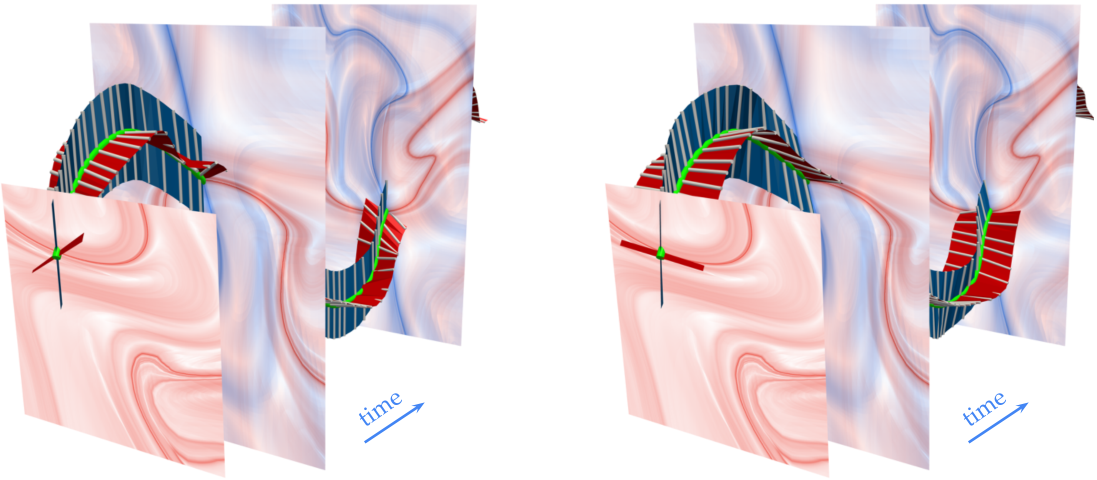
$$\frac{d}{dt}\mathbf{X}(t) = \mathbf{J}(t)\mathbf{X}(t), \quad \mathbf{X}(t_0) = \mathbb{I}, \tag{5.11}$$

where we linearly interpolate $\mathbf{J}_i$ between the discrete time steps $t_0, \dots, t_N$ to obtain $\mathbf{J}(t)$. To avoid numerical issues that occur for strong hyperbolicity or long integration times we use the continuous SVD method ([DE08], Appendix B.2) to obtain the matrices $\mathbf{B}_{t_0}(t_i)$, $\mathbf{\Sigma}_{t_0}(t_i)$, $\mathbf{R}_{t_0}(t_i)$ for $i = 0, \dots, N$. The coordinate transformations $\mathbf{T}(t_i)$ as well as $\mathbf{D}_{t_0}(t_N)$ are then computed according to Equation 5.3, and the inverse transformation $\mathbf{T}(t)^{-1}$ is obtained by inverting the factors in Equation 5.3, such that no numerical matrix inversion is involved. In cases, where $\vartheta = (t_N - t_0)\max(\mathbf{D}_{t_0}(t_N))$ exceeds the floating-point precision available for numerical computation, the factors $\exp((t_i - t_0)\mathbf{D}_{t_0}(t_N))$ and $\exp(-\mathbf{\Sigma}_{t_0}(t_i))$ usually cause catastrophic cancellation in the computation of $\mathbf{T}(t_i)$. For computation using double precision, we split $[t_0, t_N]$ into time intervals of length $T = (t_N - t_0) \cdot 15/\vartheta$. On each of these non-overlapping intervals, as well as on intervals shifted by $T/2$, we compute a refined DHT. The overlapping results are averaged, thus ensuring a uniform precision along the DHT. The refinement toward the DHT is obtained as the solution of the implicit integral Equations 5.6, which we solve using a fixed-point iteration. Starting with initial guess $\mathbf{w}^{(0)}(t) = \mathbf{0}$, we compute $\mathbf{h}(\mathbf{w}^{(0)}(t), t)$ according to Equation 5.5, and evaluate the integrals using trapezoidal rule, yielding an approximate solution $\mathbf{w}^{(1)}(t)$ of Equation 5.6. This process is iterated until $\|\mathbf{w}^{(j+1)} - \mathbf{w}^{(j)}\|$ drops below a predefined threshold τ_f or a maximum number of iterations is reached. A refined DHT is then obtained as $\tilde{\mathbf{x}}_{\mathrm{DHT}}(t)$ from Equation 5.8. Since the localization of the initial path $\tilde{\mathbf{x}}(t)$ is fixed, we use $\tilde{\mathbf{x}}_{\mathrm{DHT}}(t)$ as new initial candidate, and repeat the entire process. The iteration terminates when $\|\tilde{\mathbf{x}}_{\mathrm{DHT}}^{(j+1)} - \tilde{\mathbf{x}}_{\mathrm{DHT}}^{(j)}\|$ reaches a predefined threshold τ_i. See Figure 5.11d for an example, where two iterations are needed for convergence.

5.2.3 Streak Manifold Seeding

The attracting and repelling manifolds of the DHTs, i.e., the LCS, are extracted by computing streak manifolds, offset in the perturbation directions $\boldsymbol{\xi}_i$ that belong to the respective Lyapunov exponents λ_i. These directions yield linear approximations of the corresponding LCS and are obtained as the columns of $\mathbf{T}(t)^{-1}$. Previous work [MBES16; ÜSE13] used the real eigenvectors of the Jacobian as an estimate, which, due to their instantaneous nature, are in general not well aligned with the manifolds (Figure 5.3). In general, DHTs only exist over finite time intervals, which are typically shorter than the time domain. As shown by Üffinger et al. [ÜSE13], streak manifold integration has to continue across the entire space-time domain, however.

There are two approaches to seeding streak manifolds along a DHT. Since our DHTs are by construction hyperbolic everywhere, streak manifolds can be seeded along the entire length, i.e., by taking all offsets shown in Figure 5.3b. This approach leads to gaps in the resulting manifolds that correspond to the seeding offsets. In the 2D case, this is consistent with the seeding approach for critical points in steady VFT, where streamlines are started at offsets from the saddle-type critical points, and we thus employ this seeding method in our 2D examples. Specifically, we compute space-time streamsurfaces seeded from the polylines $\mathbf{x}(t) \pm \delta^{\pm}\boldsymbol{\xi}_2(t)$ in forward-time to obtain the repelling manifold. The vectors $\boldsymbol{\xi}_i(t)$ need to be oriented consistently along the DHT. The attracting manifold is analogously computed in backward-time offset along $\boldsymbol{\xi}_1(t)$.



(a) (b)

Figure 5.3: Space-time seeding of repelling (red) and attracting (blue) streak manifolds along a 2D DHT (green), here with exaggerated offset (white lines) for illustration purposes. (a) Offset in direction of the real eigenvectors of $\nabla \mathbf{u}(\mathbf{x}(t), t)$ deviates from the corresponding attracting LCS (blue FTLE ridges) and repelling LCS (red FTLE ridges). (b) Seeding along the Lyapunov vectors $\boldsymbol{\xi}_i = \mathbf{T}(t)^{-1}\mathbf{e}_i$, on the other hand, is approximately tangential to the respective LCS.

Since a DHT is a pathline, it is enough to construct a segment in repelling direction, that contains the initial point of the DHT, at the initial time of the DHT, and advect it in forward-time. For attracting manifolds, the final time of the DHT is used instead. Since the manifolds are attracting in their respective integration directions, the advected segment is confined to the manifold and intersects the DHT. As we are going to discuss in Section 5.3, in the case of hyperbolic path surfaces in 3D, only this seeding method is consistent with that of a 3D saddle-type critical point from steady VFT, and therefore we employ this method in the case of isolated 3D DHTs (see Figure 5.4).

In the 3D case, one-dimensional repelling streak manifolds, i.e., in the case $\lambda_2 < 0 < \lambda_3$, are seeded from the line segment $\mathbf{x}(t_0) \pm \delta^{\pm}\boldsymbol{\xi}_3(t_0)$. From this polyline, a space-time streamsurface is computed. In the case $\lambda_1 < 0 < \lambda_2 < \lambda_3$, where the repelling manifold is two-dimensional, the disc bounded by $\mathbf{x}(t_0) + \delta \sin(\alpha)\boldsymbol{\xi}_3(t_0) + \delta \cos(\alpha)\boldsymbol{\xi}_2(t_0)$, $\alpha \in [-\pi, \pi]$ is used for seeding a time surface. Attracting manifolds are extracted analogously, seeded from the end of the time interval t_N, and by replacing $\boldsymbol{\xi}_3$ with $\boldsymbol{\xi}_1$.

Streak Integration

We compute the streaklines and timelines, seeded from the seeding segments as above, as streamsurfaces [Hul92] in the space-time vector field. From these streamsurfaces,

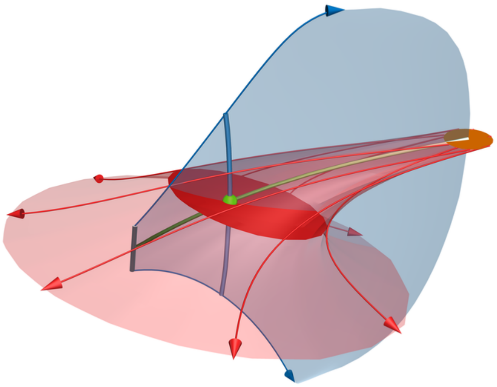


Figure 5.4: Seeding of streak manifolds at an isolated 3D hyperbolic trajectory (green). In the 4D space-time flow, forward-time stream surface integration of the 1D attracting manifold (blue, transparent), and backward-time stream volume integration of the 2D repelling manifold (red, transparent), yield manifolds in 4D, such that time slices (opaque) represent streak manifolds.

timelines and streaklines at a point in time t are obtained as the isocontour at isovalue t of the time component stored in the coordinates of the nodes of the surface mesh (see also Sections 2.4.4 and 2.4.5).

The streak surfaces and time surfaces could be computed similarly to stream volumes in the space-time vector field, where isosurfaces of the time scalar would yield the streak surfaces. However, we found that this leads to prohibitively large four-dimensional tetrahedral meshes for long integration times in turbulent flows. Therefore, the streak surfaces are computed using a minor modification of the time surface algorithm by Krishnan et al. [KGJ09]. Starting from the smallest time value in the space-time seeding mesh, we iteratively step forward in time. At each time step t, those vertices in the space-time mesh that have a time value below t are advected toward the current time step t, using a dynamically sized integration step. The remaining parts of the algorithm are performed as described in the original work, which results in a set of pathlines and a representation of the streak surface across the entire integration time. The resulting streak surfaces at time t, which may contain vertices at times greater or equal to t, are clipped by time.

5.2.4 Determination of Seeding Lengths

Typically, the lengths δ^-, δ^+, and δ are user-defined and depend on the dataset as well as the available time interval for integration. Branicki and Wiggins [BW10, Appendix B] have shown that for sufficiently small choices, the resulting manifolds are guaranteed to be contained within the respective finite-time invariant manifolds, and that the influence of this choice diminishes with increasing advection time. However, numerical datasets typically have limited time domains, making it important to select a length that is as large as possible, to ensure that the streak manifolds grow large enough in the

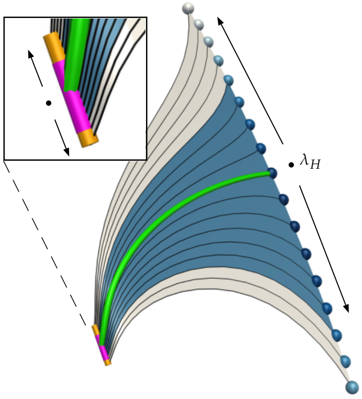


Figure 5.5: Determination of an optimal seeding length δ. A space-time streak surface (white) is computed from an initially large offset ($\delta_{\max}$, orange). At its end points, the localized FTLE values in reverse time (spheres, white to blue) in relation to the FTLE λ_H of the HT are used to determine (along arrows) the seeding offset (magenta).

available space-time domain of the dataset and the lifetime of the HT. We determine such a length using the following approach (see Figure 5.5). From a user-defined maximum length $\delta_{\max}$, we compute the resulting streakline as a space-time streamsurface over the time interval of the hyperbolic trajectory, where we obtain the set of pathlines that are involved in its computation. If a new seed is inserted during the adaptive extraction [Hul92], we linearly interpolate the two involved pathline seeds, instead of the triangle edge, to obtain pathlines over the entire time interval. This results in a mapping between points on the regularly sampled streakline and its initial points on the seeding segment. Finally, starting from the hyperbolic trajectory, we determine the closest seeding point, such that the reverse-time localized FTLE [Kas+09] of the corresponding point on the streakline falls below a percentage of 50% of the FTLE λ_H at the HT, for which we obtained good results in our experiments. This approach is demonstrated, using varying percentages, in Figures 5.31b to 5.31d.

5.3 Extraction of Hyperbolic Path Surfaces

Following the discussion in Section 5.1.2, in 3D time-dependent flows, we additionally extract hyperbolic path surfaces by refinement of candidate surfaces. Figure 5.6 provides an overview of the method.

5.3.1 Candidate Extraction

Taking an instantaneous view of hyperbolic path surfaces in the steady 4D space-time vector field $\bar{\mathbf{u}}(\bar{\mathbf{x}}) := (\mathbf{u}(\mathbf{x},t),1)^\top, \bar{\mathbf{x}} := (\mathbf{x},t)^\top$, they can be seen as bifurcation surfaces (Section 3.1.4), i.e., locations where the space-time Jacobian $\nabla\bar{\mathbf{u}}(\bar{\mathbf{x}})$ has only real eigenvalues, and $\bar{\mathbf{u}}(\bar{\mathbf{x}})$ lies in the plane spanned by its two eigenvectors $\bar{\boldsymbol{\eta}}_2(\bar{\mathbf{x}}), \bar{\boldsymbol{\eta}}_3(\bar{\mathbf{x}})$,

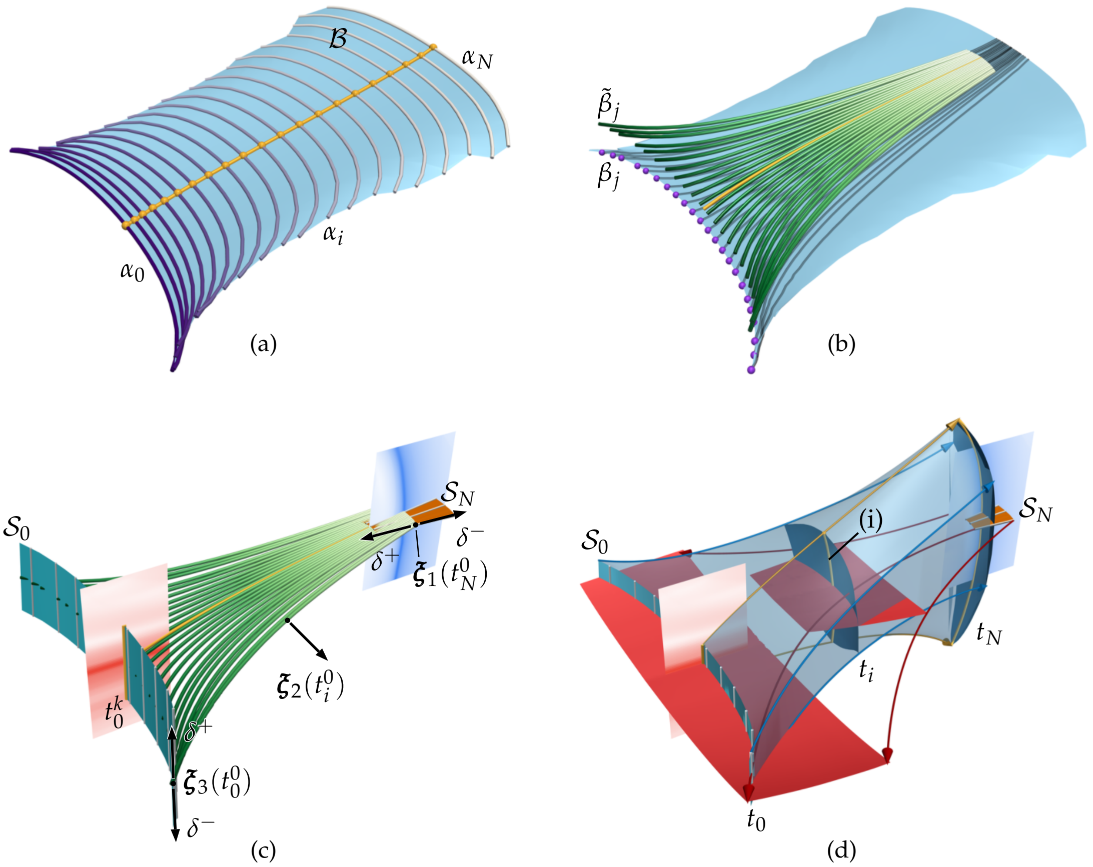


Figure 5.6: Extraction of 3D time-dependent vector field topology: (a) Bifurcation lines α_i (time purple to white) are extracted in $\mathbf{w}(\mathbf{x}, t)$ at each time step and triangulated ($\mathcal{B}$) in space-time. (b) Streamlines of $\bar{\mathbf{u}}(\bar{\mathbf{x}})$ constrained to $\mathcal{B}$ (β_j, black, seeded at spheres) are refined toward hyperbolic trajectories ($\tilde{\beta}_j$, green to white) within the hyperbolic path surface. (c) The major ($\boldsymbol{\xi}_1$) and minor ($\boldsymbol{\xi}_3$) Lyapunov vectors at the end points of each hyperbolic trajectory are approximately perpendicular to the respective FTLE ridge, and used as seeding directions (white tubes, shown exaggerated), resulting in seeding manifolds $\mathcal{S}_0$ (cyan), $\mathcal{S}_N$ (orange). The medium Lyapunov vectors $\boldsymbol{\xi}_2$ are approximately tangential to the hyperbolic path surface. (d) Space-time streamvolumes (transparent blue, forward shown only) are seeded at $\mathcal{S}_0$, $\mathcal{S}_N$, whose time-slices are streak surfaces (opaque red and blue). A straightforward extension of the 2D method would refine paths of saddle-type critical points ((a), yellow) toward hyperbolic trajectories ((b),(c), yellow). The resulting streak line ((d)(i), yellow) misses most of the surface-type LCS ((d), blue).

that belong to its two medium eigenvalues $\mu_2(\bar{\mathbf{x}}) \leq \mu_3(\bar{\mathbf{x}})$. This is a direct extension of the concept of swirling particle cores [WSTH07], which employs the reduced velocity criterion [SH95]. Similarly, since the space-time Jacobian has exactly one zero eigenvalue, whose eigenvector is the feature flow field $\mathbf{f} = -\nabla\mathbf{u}^{-1}\mathbf{u}_t$, this definition is reduced to moving 3D bifurcation lines in the Galilean-invariant frame of reference

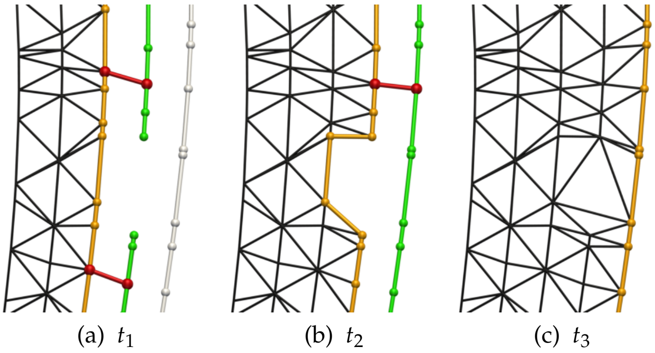


Figure 5.7: Three consecutive time steps in the space-time triangulation of parallel vectors lines. A front (orange) of polylines is triangulated greedily with polylines in the candidate set (green), started from the pairs of closest vertices (red).

$\mathbf{w}(\mathbf{x},t) = \mathbf{u}(\mathbf{x},t) - \mathbf{f}(\mathbf{x},t)$. As described in Section 5.2.1 for the extraction of DHTs, we compute this frame of reference by solving a minimization problem (Equation 5.10).

Since the observed vector fields in two neighboring time steps $\mathbf{w}(\mathbf{x},t_1)$, $\mathbf{w}(\mathbf{x},t_2)$ belong to different observers, we found that, for numerical datasets, using temporal derivatives of $\mathbf{w}(\mathbf{x},t)$ for constructing a feature flow field for parallel vectors lines [The+05], or temporal interpolation for tracking solutions in space-time grid cells [BP02] leads to unreliable and noisy results. Similarly, applying our dependent vectors operator (Chapter 3) to solve $\bar{\mathbf{u}}(\bar{\mathbf{x}}) \wedge \bar{\boldsymbol{\eta}}_2(\bar{\mathbf{x}}) \wedge \bar{\boldsymbol{\eta}}_3(\bar{\mathbf{x}}) = \mathbf{0}$ directly leads to prohibitively noisy results, as demonstrated in Section 3.3.4.

Instead, we extract bifurcation lines in the 3D spatial domain by applying the parallel vectors operator to each time step, and filter the resulting lines by feature strength and length [MSE13]. Note that unlike the triangulated space-time surfaces, the bifurcation lines can be filtered reliably (cf. Figures 3.5d and 3.5e). The resulting sequence of polylines is triangulated in space-time using an approach based on the ball-pivoting algorithm [Ber+99] for surface reconstruction of 3D point clouds. A similar approach was employed by Wilde et al. [WRT18b], who performed triangulation of a 5D point cloud in its 3D projection. Note, however, that a straightforward extension of the ball-pivoting algorithm to 4D would only be possible for reconstructing volumes, since it relies on (hyper-)surface normals. Similarly to the original algorithm [Ber+99], we perform the triangulation by advancing a front of edges. In each iteration, we insert the polylines of the next time step into a set of candidates for triangulation. Edges in the front are greedily advanced to candidate edges if they are closer than a cell diagonal, creating new triangles. At the end of each iteration, the remaining candidates are added to the front, and edges, which belong to time steps older than two, are removed from the front. See Figure 5.7 for an illustration of the procedure.

Subsequently, we extract the paths of saddle-type critical points in $\mathbf{w}(\mathbf{x}, t)$ to obtain further candidates for DHTs (Section 5.2.1). Those paths that are not contained in the obtained triangulation are added to the set of candidate manifolds, and the remaining ones are discarded. These additional paths address isotropic saddle-type (Figure 5.2c) and spiral saddle-type critical points (Figures 5.14a and 5.14e).

5.3.2 Refinement

The candidate surfaces from the previous section are first decomposed into an ordered set of candidate lines. For this, we integrate the space-time vector field $\bar{\mathbf{u}}(\bar{\mathbf{x}})$ projected onto the surface. The projected 4D vector field is converted into a piecewise constant vector field on each triangle. We then seed a streamline at the center of each edge on the boundary of the candidate surface, and integrate it according to Tricoche at al. [TGS06]. If the projected velocity points outside of the triangle at the boundary edge, we use backward-time integration. By construction of the space-time surface, streamline integration starts and ends on the surface boundary. This allows us to order the resulting candidate lines. Figure 5.6b shows an example.

We then refine each of the candidate lines using the method for computing DHTs from Section 5.2.2. Note that by continuity of the refinement scheme sets of candidate lines that span surfaces will remain surfaces after refinement, and generally do not collapse to single lines (as would be suggested by the attribute "distinguished"; see Sections 5.1.1 and 5.1.2 for a discussion). This computation also provides the Lyapunov vectors along each of the obtained pathlines, which we, again, use for seeding streak manifolds. The method for obtaining optimal seeding lengths (Section 5.2.4) is applied for each HT separately, resulting in varying lengths along the HPS.

5.3.3 Streak Manifold Seeding

Streak manifold seeding for single 3D hyperbolic trajectories was discussed in Section 5.2.3. Recall that a 3D DHT possesses a pair of a 1D and a 2D streak manifold. In practice, we need to generate a streak manifold from a hyperbolic path surface instead, with the HPS given (Section 5.3.2) as a sorted set of hyperbolic trajectories $\{\mathbf{x}^j(t)\}$, where each can be considered to possess a seeding structure as described in Section 5.2.3. Furthermore, all Lyapunov exponents λ_i^j are of equal sign for fixed i, and the HPS is approximately tangential to the medium Lyapunov vectors $\boldsymbol{\xi}_2^j(t)$. For seeding the repelling manifold, we construct a space-time triangle mesh $\mathcal{S}_0$ by triangulating the family of line segments $\mathbf{x}^j(t_0^j) + \delta_j^\pm \cdot \boldsymbol{\xi}_3^j(t_0^j)$ (see Figure 5.8). Note that the

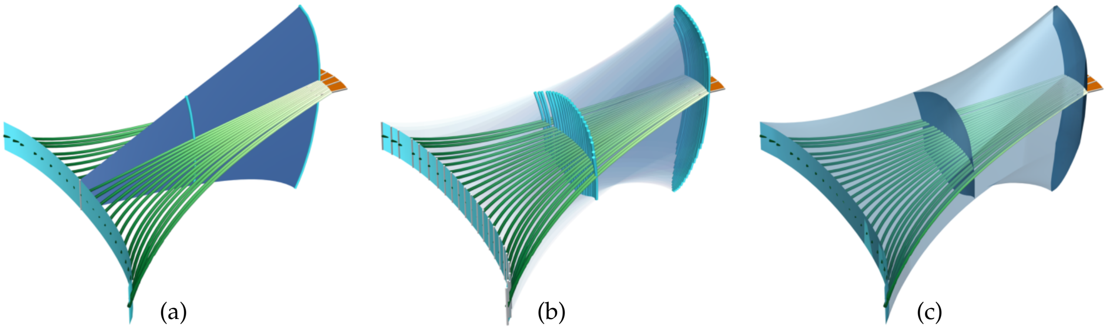


Figure 5.8: Each HT (green) belonging to a HPS can be thought of possessing, in addition to a 2D steak manifold, a 1D streak manifold (a). Their union (b) is a streak surface, which can be extracted as space-time stream volume seeded from the triangulation of seeding line segments (c).

initial times t_0^j are in general not constant. Analogously, also a 2-manifold seeding structure in space-time is obtained for the attracting manifold ($\mathcal{S}_N$ in Figure 5.6c).

5.4 Properties of Time-Dependent Vector Field Topology

In this section, we discuss general properties of time-dependent vector field topology, using our method for construction of illustrative examples.

5.4.1 Consistency with Steady Vector Field Topology

VFT of steady vector fields is known to coincide with the topology indicated by ridges in the FTLE fields, in the limit as advection time tends to infinity [FGRT17]. We further note that for steady 3D vector fields the medium Lyapunov exponent λ_2 is zero, since the medium Lyapunov vector $\boldsymbol{\xi}_2$ describes perturbation in direction of the flow $\boldsymbol{\xi}_2(t) \parallel \mathbf{u}(\mathbf{x}(t), t)$, and therefore HPSs degenerate to points or lines. We thus verify in the following that the structures extracted from a steady vector field $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_s(\mathbf{x})$ by our approach coincide with saddle-type critical points and bifurcation lines, and their invariant manifolds in the steady vector field $\mathbf{u}_s(\mathbf{x})$.

A steady vector field has zero time-derivative $\mathbf{u}_t(\mathbf{x}, t) = \mathbf{0}$, and thus any reference frame, which minimizes a time derivative, coincides with the vector field itself, i.e., $\mathbf{w}(\mathbf{x}, t) = \mathbf{u}_s(\mathbf{x})$ and $\mathbf{f}(\mathbf{x}, t) = \mathbf{0}$. Saddle-type critical points, which, in the 3D case, do not lie on bifurcation lines, are thus extracted as temporal candidate lines ($\mathbf{x}(t) = \text{const}$), and the subsequent refinement does not change their location, since they are already hyperbolic trajectories. In 3D, candidate surfaces consist of temporal copies of the bifurcation lines in $\mathbf{u}_s(\mathbf{x})$. These are decomposed into surface streamlines (Section 5.3.2), which, again, represent copies of the bifurcation lines in $\mathbf{u}_s(\mathbf{x})$ with an additional time-parametrization. Refinement of these candidate lines toward hyperbolic trajectories is equivalent to refining the steady bifurcation lines toward streamlines. This process is similar to the technique of Machado et al. [MBES16; MSE13], who use the same algorithm for extraction of 3D bifurcation lines and 2D hyperbolic trajectories in space-time. Thus, our seeding manifolds coincide with the usual seeding approaches for saddle-type critical points and bifurcation lines in traditional VFT. Our concept is consistent with VFT, since streak surfaces and streamsurfaces are identical for steady vector fields. Non-hyperbolic features from steady VFT, such as source- or sink-type critical points and periodic orbits as well as invariant tori, do not have to be considered in this context, as we are going to discuss next.

5.4.2 Desirable Properties for Unsteady Topology

We now discuss desirable properties for unsteady topology, as proposed by Bujack et al. [Buj+20]. According to the authors, properties required for physical meaningfulness are: coincidence with the steady flow topology, induction of a partition of the domain, Lagrangian invariance, objectivity, and Galilean invariance.

Our method is consistent with the subset of steady vector field topology (Section 5.4.1) defined by saddle-type critical points and bifurcation lines. As we only consider hyperbolic structures, consistent with LCS, the missed features in steady VFT are those that do not contribute to hyperbolic separation. The spatial domain is only partially partitioned due to time-dependence, and since LCS and FTLE ridges need sufficient time to grow. This also implies that Lagrangian invariance and a non-trivial spatial partition are mutually exclusive in general. Most non-trivial partitions are induced by intersections of the streak manifolds themselves, i.e., at the presence of unsteady saddle connectors (Section 5.4.5). Our topology is Lagrangian-invariant, as it is defined by pathlines and streak surfaces, and thus also Galilean-invariant and objective. Note that even the extraction using initial candidates in a possibly not even Galilean-invariant frame of reference would yield objective topological features in our approach, albeit some initial candidates could possibly be missed. We refer to Branicki and Wiggins [BW10, Sec. 2] for an in-depth discussion.

5.4.3 FTLE and Separation in Steady Vector Fields

Bounded ABC Flow. The ABC flow with standard parameters $A = \sqrt{3}, B = \sqrt{2}, C = 1$ is a steady 3D analytical solution of Euler's equation, which is commonly used as a

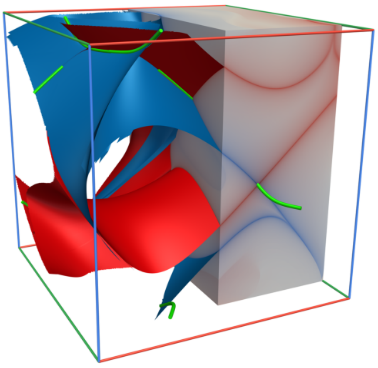


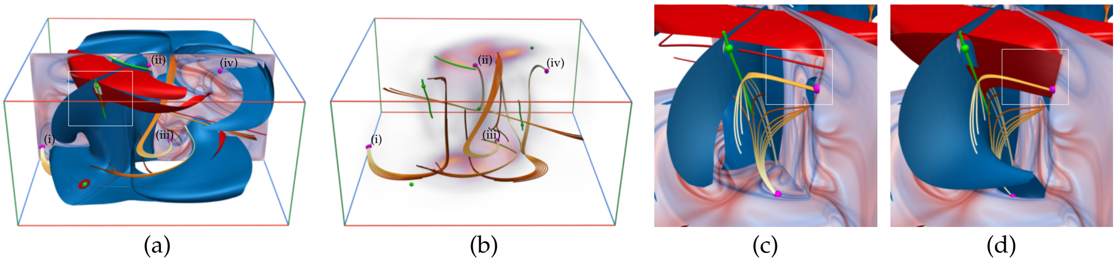
Figure 5.9: While traditional VFT in the Bounded ABC Flow is empty, bifurcation lines (green) and their invariant manifolds (red/blue) capture the separating behavior of the flow consistently with the FTLE (box, red/blue color).

benchmark for Lagrangian analysis [Hal01]. Generally, periodic spatial boundaries with fundamental domain $[0, 2\pi]^3$ are assumed. However, when regarding this steady vector field as bounded, as is common for numerical datasets, it exhibits no VFT structures, i.e., no critical points, nor periodic orbits or invariant tori. The FTLE, on the other hand, exhibits ridges, which separate the domain into regions of similar flow behavior. Separatrices of bifurcation lines also extract these structures (Figure 5.9), which motivates their inclusion in VFT and thus in our time-dependent topology.

Instantaneous 3D Convective Flow. We consider the instantaneous topology of a single time step of the 3D Convective Flow dataset (Section 5.6.4), and compare it with FTLE ridges (Figure 5.10a–5.10d). For this, we integrate the invariant manifolds of the VFT using the same integration time as the FTLE. In this dataset, the FTLE exhibits many more ridges than captured by the separatrices (Figure 5.10a). Investigating their separating behavior by seeding rakes across these ridges (Figure 5.10b), we find that ridges not captured by VFT are shear-induced. We conclude that employing the FTLE for flow analysis requires careful examination of the separating behavior of each ridge. The dataset further contains saddle-type critical points, which lie on bifurcation lines. Comparing the corresponding invariant manifolds, which can be either seeded from the critical point (Figure 5.10c) or the bifurcation line (Figure 5.10d), we find that seeding from the bifurcation line results in topological structures more consistent with the FTLE. For this reason, our unsteady approach favors bifurcation lines over critical points, i.e., we omit critical points lying on bifurcation lines (e.g., Figure 5.32a).

5.4.4 Local Predictability of DHTs

Since part of our DHT extraction consists of the local extraction of initial candidates, the question arises, whether the local extraction itself could be modified, such that it



(a) (b) (c) (d)

Figure 5.10: Steady vector field topology of a single time step in the 3D Convective Flow (saddle-type critical points and bifurcation lines green, invariant manifolds red and blue). Rakes (i–iv in (a),(b): seeds magenta, streamlines with integration time orange to white) show that shear-induced ridges in the FTLE fields (section) are not included in the steady VFT. Seeding invariant manifolds of a saddle-type critical point (c) instead of the bifurcation line it is contained in (d) misses parts of the FTLE ridges at same integration times (box). Enlarged region, see box in (a).

would yield the exact location of the DHT. The discussion in Section 5.1 suggests that a local extraction is not possible. We construct two 2D vector fields $\mathbf{u}^{(1)}(\mathbf{x},t)$, $\mathbf{u}^{(2)}(\mathbf{x},t)$ based on Shadden's double gyre model [SLM05],

$$u^{(i)}(x,y,t) = -\sin\left(\pi f^{(i)}(x,t)\right)\cos\left(\pi f^{(i)}(y,t)\right)\frac{df^{(i)}}{dz}(y,t), \tag{5.12}$$

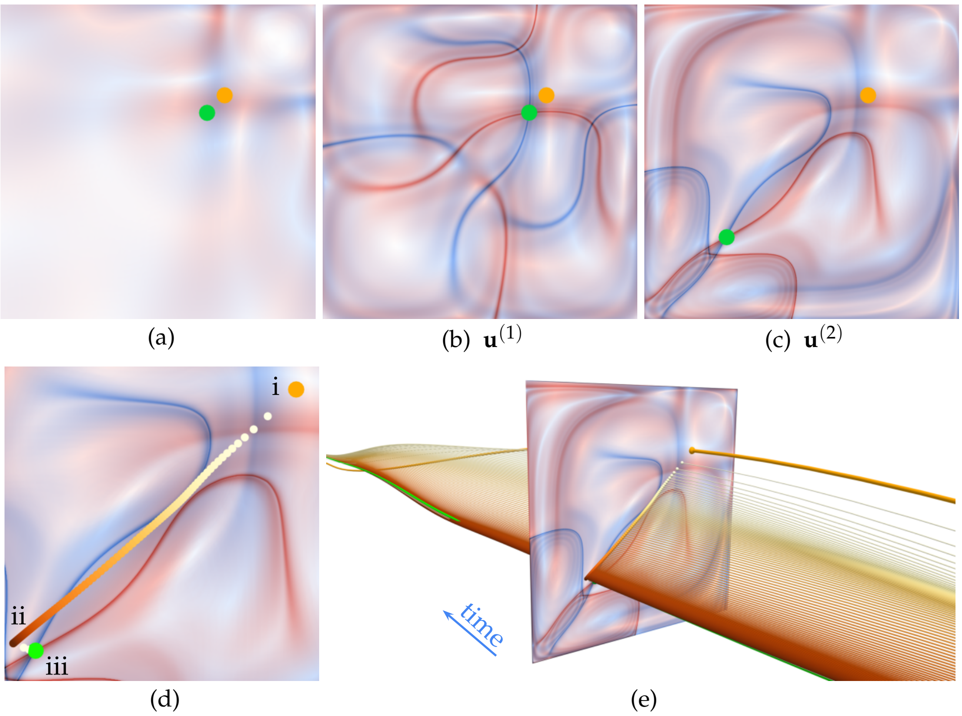
$$v^{(i)}(x,y,t) = \cos\left(\pi f^{(i)}(x,t)\right)\sin\left(\pi f^{(i)}(y,t)\right)\frac{df^{(i)}}{dz}(x,t), \tag{5.13}$$

$$f^{(i)}(z,t) = a^{(i)}(t)z^2 + \left(1-2a^{(i)}(t)\right)z, \tag{5.14}$$

$$a^{(1)}(t) = \frac{1}{2}\cos(\max(|t\pi/10|,\pi/2)), \tag{5.15}$$

$$a^{(2)}(t) = \frac{1}{2}\cos(\max(|t\pi/10|,\pi)). \tag{5.16}$$

The vector fields are defined on the spatial domain $[0,0]\times[2,2]$. They contain a saddle-type critical point at approximately $(1+a^{(i)}(t),1+a^{(i)}(t))^\top$. Starting from $t=0$, in $\mathbf{u}^{(1)}$, the critical point moves from $(1.5,1.5)^\top$ to $(1,1)^\top$ over 5 s, while the critical point in $\mathbf{u}^{(2)}$ moves to $(0.5,0.5)^\top$ over 10 s. Both fields are symmetric in time, and they coincide over the time interval $[-5\,\mathrm{s},5\,\mathrm{s}]$. Thus, any local algorithm that at most considers data on this interval would yield the same result at $t=0\,\mathrm{s}$. However, as Figure 5.11 shows, the location of the DHT, which starts to become visible using the FTLE over the time interval $[-10\,\mathrm{s},10\,\mathrm{s}]$, differs tremendously in the two flows. We therefore conclude that hyperbolic trajectories are not defined by local properties of the flow, but by the global dynamics. The convergence of the DHT refinement is shown in Figures 5.11d and 5.11e. After the first fixed-point iteration (i to ii, white to brown), the exact location is not yet reached, since the localized flow along the initial candidate (orange sphere) was used.



(a) (b) $\mathbf{u}^{(1)}$ (c) $\mathbf{u}^{(2)}$

(d) (e)

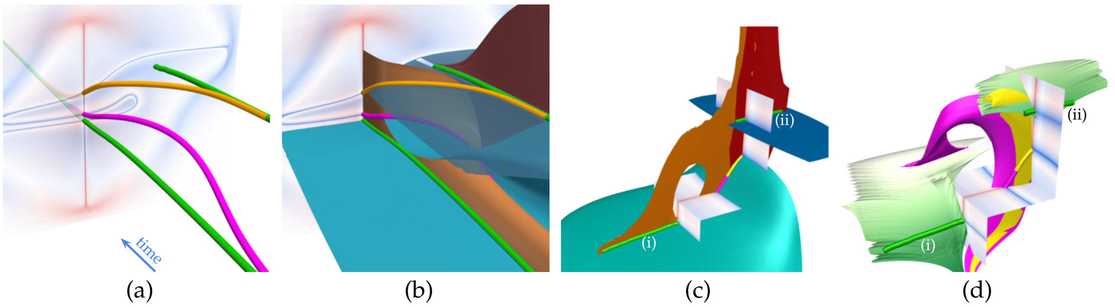
Figure 5.11: Forward and backward FTLE at t=0 s of the 2D counterexamples $\mathbf{u}^{(1)}(\mathbf{x},t)$, $\mathbf{u}^{(2)}(\mathbf{x},t)$ coincide for up to ±5 s advection time (a), but differ for advection time of 15 s (b)(c). While the location of the critical point (orange) in the displacement-invariant frame of reference [BG20], like any other local criterion, is independent of the time span considered, only at advection times greater than 5 s, the location of the DHT (green) is clearly defined. (d) Started from the path of the critical point in $\mathbf{u}^{(2)}(\mathbf{x},t)$, the DHT refinement (two iterations, i to ii to iii) reaches the FTLE ridge intersection, since it incorporates flow information along the entire interval (e).

The result of this is used to recompute the localized flow and start a second fixed-point iteration in our scheme (ii to iii), which quickly reaches the intersection of FTLE ridges.

5.4.5 Unsteady Saddle Connectors

In 3D steady vector fields, the separatrices of saddle-type critical points can intersect in streamlines, which form connections between two different (heteroclinic) or one (homoclinic) saddle-type critical point. These are also called saddle connectors ([TWHS03], see also Section 2.6.4). As we have seen, saddle connectors in 4D steady vector fields can be surfaces (Section 4.6.2). By considering the space-time flows of 2D and 3D time-dependent vector fields, which are 3D and 4D, respectively, we obtain an analogy.

Similar connections can be formed by hyperbolic trajectories in unsteady vector fields [MW98]. They are typically formed at a time t_i by manifolds of DHTs, which



(a) (b) (c) (d)

Figure 5.12: Unsteady saddle connectors: heteroclinic (yellow) and homoclinic (magenta) orbits between two 2D DHTs (a),(b) and 3D HPSs ((c),(d), (i),(ii), green), in a single time step (c) and time projection (a),(c),(d) including the connectors. The repelling (orange/red) and attracting (cyan/blue) manifolds intersect along lines in 3D space-time (a),(b), and along surfaces in 4D space-time (c),(d). Ridges in the forward- and backward-time FTLE fields (red/blue color mapped surfaces) intersect both along DHTs or HPSs, as well as unsteady saddle connectors.

have come into existence at a time in the past ($t < t_i$) or in the future ($t > t_i$). For sufficiently long advection time, they can be observed as intersections of ridges in the forward and backward FTLE fields, which, however, do not represent DHTs or HPSs, since they are not generators of the topology (see Figure 5.12).

For illustration, we construct an analytic 2D example, from two stationary saddle-type linear fields, with instantaneous critical points at $(0,0)^\top$ and $(1,1)^\top$. Both are made nonlinear with Gaussian window functions. The saddle at $(1,1)^\top$ is faded into and out of existence over a short time period, resulting in a short DHT and one DHT that exists for all times. The DHTs and their manifolds are shown in Figures 5.12a and 5.12b. The intersection curves of their manifolds are visible in the forward and backward FTLE fields computed at a time slice, where the shorter DHT does not exist. These unsteady equivalents of saddle connectors can be hyperbolic trajectories, but they do not need to be. However, they are not DHTs, since they are not generators of the time-dependent topology of the flow. Since they arise from non-local mechanisms, however, local extraction methods (such as ours) usually do not yield candidate lines for these false-positives (cf. Figure 5.21). This example is extended to 3D in Figures 5.12c and 5.12d, such that it exhibits a surface-type unsteady saddle connector.

5.4.6 Scale-Dependency of the FTLE

The separation that induces an FTLE ridge is defined by its curvature [SLM05]. This makes analysis using FTLE ridges even more challenging, since the curvature depends on the spatial scale/resolution of the FTLE field. The choice of resolution implies a

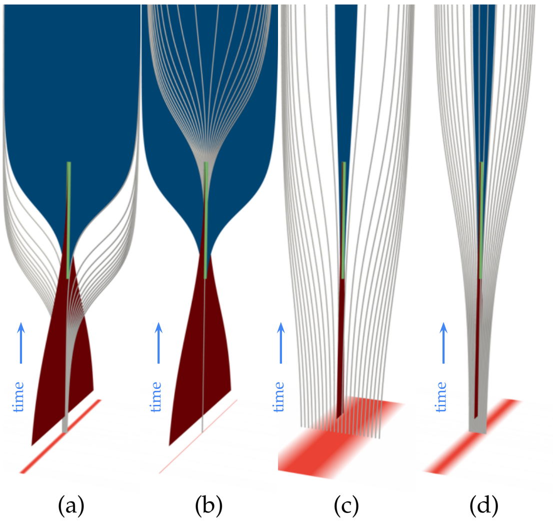


Figure 5.13: 2D flows exemplifying the scale-dependency of the FTLE (red, bottom). Strongly separating LCS (a) appears weak at smaller scales (b). A weakly separating LCS (c) appears strong at larger scales (d). Hyperbolic trajectory (green) and attracting/repelling streak manifolds (blue/red).

choice of scale, as we demonstrate at two simple 2D examples in Figure 5.13. These examples contain a small hyperbolic region, where we extract a hyperbolic trajectory, of varying strength (large in Figures 5.13a and 5.13b, small in Figures 5.13c and 5.13d). In both cases, the corresponding FTLE ridge can be extracted with high or low curvature, depending on FTLE resolution. Our locally extracted topology, on the other hand, has no such scale parameter. Instead, the resulting streak manifolds exhibit different levels of growth. In practice (see also Section 5.6.4), hyperbolic trajectories can be filtered by strength in order to obtain only the most relevant topological structures.

5.5 Analysis

We now turn to the analysis of the convergence and performance properties of our approach, using analytic examples as well as the numerical datasets, which will be discussed and evaluated in detail in Section 5.6. Ridge surface extraction was performed using the prototype provided by Schultz et al. [STS09].

5.5.1 Kinematic Test Cases

We construct a set of simple test cases by translating a steady vector field along a curve, i.e., the unsteadiness is of kinematic rather than dynamic nature. Such synthetic examples are commonly used as benchmark for unsteady topology [BW09; SW10; ÜSE13]. Note that this translation does not correspond to a reference frame transformation of the steady vector field.

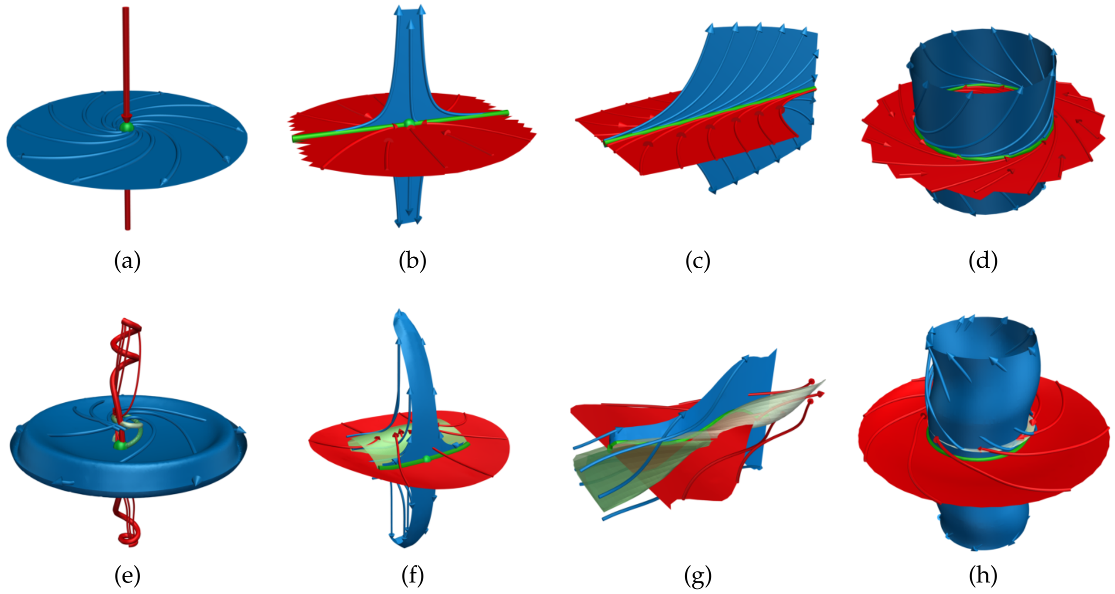


Figure 5.14: Steady 3D topological structures (top row), and their unsteady counterparts (bottom row). Spiral saddle-type critical point (a)(e), saddle-type critical point on a bifurcation line (b)(f), bifurcation line without critical point (c)(g), and closed bifurcation line (d)(h) representing a saddle-type periodic orbit. Green: hyperbolic trajectories and hyperbolic path surfaces, blue/red surfaces: attracting/repelling LCS, blue/red lines: streamlines and pathlines.

2D Skewing Osciallating Gyre-Saddle. Based on the 2D Skewing Gyre-Saddle and Oscillating Gyre-Saddle examples proposed by Sadlo and Weiskopf [SW10], we construct a model for a hyperbolic region, which oscillates sinusoidally between $(0.2, 0.8)^\top$ and $(-0.2, -0.8)^\top$ with a period of 4 s, with the saddle directions skewing at a period of 1 s. Figures 5.1 and 5.3 show this dataset over the time interval $[0, 4]$. While this dataset is time-periodic and thus would allow to compute the FTLE beyond the time boundaries, we choose not to do so. Since we only extract the DHT and its attracting and repelling manifolds over this time interval, the resulting topology will only be consistent with ridges in the FTLE fields on the same time interval. For example, the forward FTLE at t=0 s (see Figure 5.1, front slice) can thus only constrain the DHT to a line at this instant of time, while the forward and backward FTLE fields at the center of the time interval (t=2 s) constrains it to the point, where forward and backward FTLE ridges intersect. This is consistent with the computation of streak manifolds (Figure 5.1).

3D Topological Building Blocks. Starting from 3D steady configurations of topological structures, we construct their 3D unsteady counterparts, as captured by our unsteady topology, by translation along a Lissajous curve [ÜSE13]. Figure 5.14 shows the

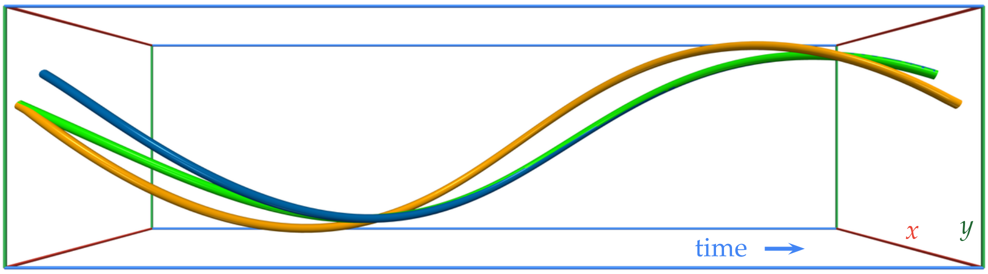


Figure 5.15: The attractor (blue) in the analytic model of the Beads problem can be found by integrating in forward time from an initial candidate line (orange) to obtain an approximation (green). The quality of the approximate solution increases with length of integration time.

constructed cases, which provide the building blocks for 3D unsteady vector field topology. Instantaneous saddle-type critical points, which do not lie on a bifurcation line, are typically found in regions, where the Jacobian exhibits complex eigenvalues, since in that case two of their real parts are equal in a numerically stable way (Figure 5.14a and 5.14e; see also Figure 5.2). Bifurcation lines can either be open (Figure 5.14c and 5.14g) or closed (Figure 5.14d and 5.14h), and they may contain a saddle-type critical point (Figure 5.14b and 5.14f). As discussed in the previous section, we favor bifurcation lines over critical points if both are present. Note that bifurcation lines do not extract general saddle-type periodic orbits, but only those that entirely represent bifurcation lines [MSE13]. Those parts of the hyperbolic path surface, on which pathlines have enough time to recirculate, coincide with recirculation surfaces [WRT18b].

5.5.2 The Beads Problem

The Beads problem is an example for an attractor (Figure 5.15), around which pathlines exhibit swirling motion. In the following, we use the analytic model and ground truth previously employed by Weinkauf and Theisel [WT10]. While traditional feature extraction methods fail to find this attractor, it can be extracted from particle density estimation [Wie+11], using streakline cores [WT10], or as rotational invariant vortex core [GST16]. This attractor can be regarded as a “sink-type DHT” [BDZG19], as it has only negative Lyapunov exponents. Note that the definition of a DHT due to Ide et al. [ISW02] also includes this case, since their notion of hyperbolicity means non-zero Lyapunov exponents rather than saddle-like behavior. We therefore may refine the erroneous parallel vectors solution toward the attractor using the algorithm described in Section 5.2.2. In this case, integrating the decoupled system (Equation 5.4) is equivalent to integration of the original flow. The initial candidate is thus increasingly refined

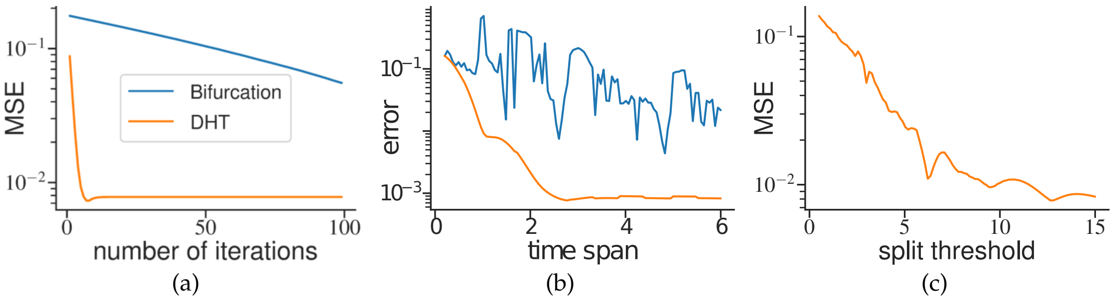


Figure 5.16: Accuracy of our method compared with the bifurcation line refinement scheme by Machado et al. [MSE13], depending on number of iterations (a), measured by mean squared error (MSE), and depending on time length of the candidate line (b), measured at the center point. (c) Accuracy of our DHT refinement for different thresholds for time interval splitting, that avoids numerical cancellation for computation with double precision, measured by MSE.

toward the ground truth with increasing time (to the right in Figure 5.15), while the point at the left time boundary is left unchanged. Integration-based methods are able to extract this attractor accurately near the beginning of the time domain, but require a dense computation of pathlines. Our method, on the other hand, is only accurate at later time steps, but only requires a localized integration.

5.5.3 Convergence of the Refinement

We use a simple 2D analytic model, for which the ground truth is known, to measure convergence of our DHT refinement, and compare it to the refinement scheme of Machado et al. [MSE13]. The time-dependent 2D vector field is defined componentwise as $u_i(\mathbf{x}, t) = d_i x_i + A_i \sin(\omega_i t)$. Since it is periodic in time, the DHT for all times can be obtained in closed form [ISW02] as

$$x_i = -A_i(d_i^2 + \omega_i^2)^{-\frac{1}{2}} \sin(\omega_i t + \arctan(\omega_i / d_i)). \tag{5.17}$$

Locations of vanishing acceleration, and thus the parallel vectors solution, are obtained, by straightforward calculation, as

$$x_i = -A_i d_i^{-2}(d_i^2 + \omega_i^2)^{\frac{1}{2}} \sin(\omega_i t + \arctan(\omega_i / d_i)). \tag{5.18}$$

To avoid a possible influence of the numerically evaluated parallel vectors operator, we use this analytic representation instead. We fix the parameters $d_1 = 3$, $d_2 = -3$, $\omega_1 = 2$, $\omega_2 = 3$, $A_1 = A_2 = 1$, and sample the analytic field on a regular grid with 100^3

nodes over the domain $[-10, 10]^2 \times [-5, 5]$. We sample the PV line at 128 equidistant instances over the time span $[-3, 3]$.

For varying numbers of iterations of our scheme, we measure the pointwise mean square error to the ground truth (Figure 5.16a). For the computation of our DHT refinement, 100 fixed point iterations, together with the continuous SVD, took 28 ms on average, while 100 iterations of the bifurcation line refinement [MSE13] took 19 ms. However, our DHT refinement converges quickly after about 20 iterations, while the bifurcation line refinement takes 2000 iterations until a good solution is obtained. The MSE for both methods stays above a rather large value. With our DHT refinement, the reason for this are the initial values at the ends of the candidate line (Equation 5.7), while the bifurcation line refinement scheme cannot distinguish between the DHT and other path lines converging toward it at the end points. The convergence in the case of a nonlinear field is shown in Section 5.4.4. Next, we measure the influence of the time length of the initial candidate on the two algorithms (Figure 5.16b). For different amounts s of time, we sample the PV line over the time interval $[-s/2, s/2]$, while keeping the sampling density equal to the previous experiment. Fixing the number of fixed point iterations at 100 for the DHT refinement, and the number of iterations for the bifurcation line refinement at 2000, we measure the distance of the center vertex to the ground truth, since both methods tend to be most inaccurate at the endpoints. Since our algorithm is integration-based, the accuracy increases smoothly with the amount of time available along the candidate line. The bifurcation line refinement scheme, on the other hand, is generally unstable. Finally, we measure the impact of splitting the time interval into intervals of $T = (t_N - t_0)s/\vartheta$ for different s. This was introduced with $s = 15$ in Section 5.2.2 to avoid numerical errors. With fluctuations due to numerical errors, accuracy increases exponentially with s (Figure 5.16c).

5.5.4 Stability under Perturbation

We analyze the stability of our DHT refinement when applied on a perturbed initial candidate, which investigates robustness against inaccurate candidate extraction, and applied on the perturbed ground truth DHT, which investigates how far away a candidate is allowed to be from the DHT for convergence. We employ two kinds of perturbations: symmetric perturbation, zero at the center, linearly increasing outward, and asymmetric perturbation with zero at the beginning, linearly increasing in forward time.

First, we use the same analytic model as in Equation 5.17 over the time span $[-2, 2]$, where the ground truth is known (Figure 5.17). The ground truth has Lyapunov expo-

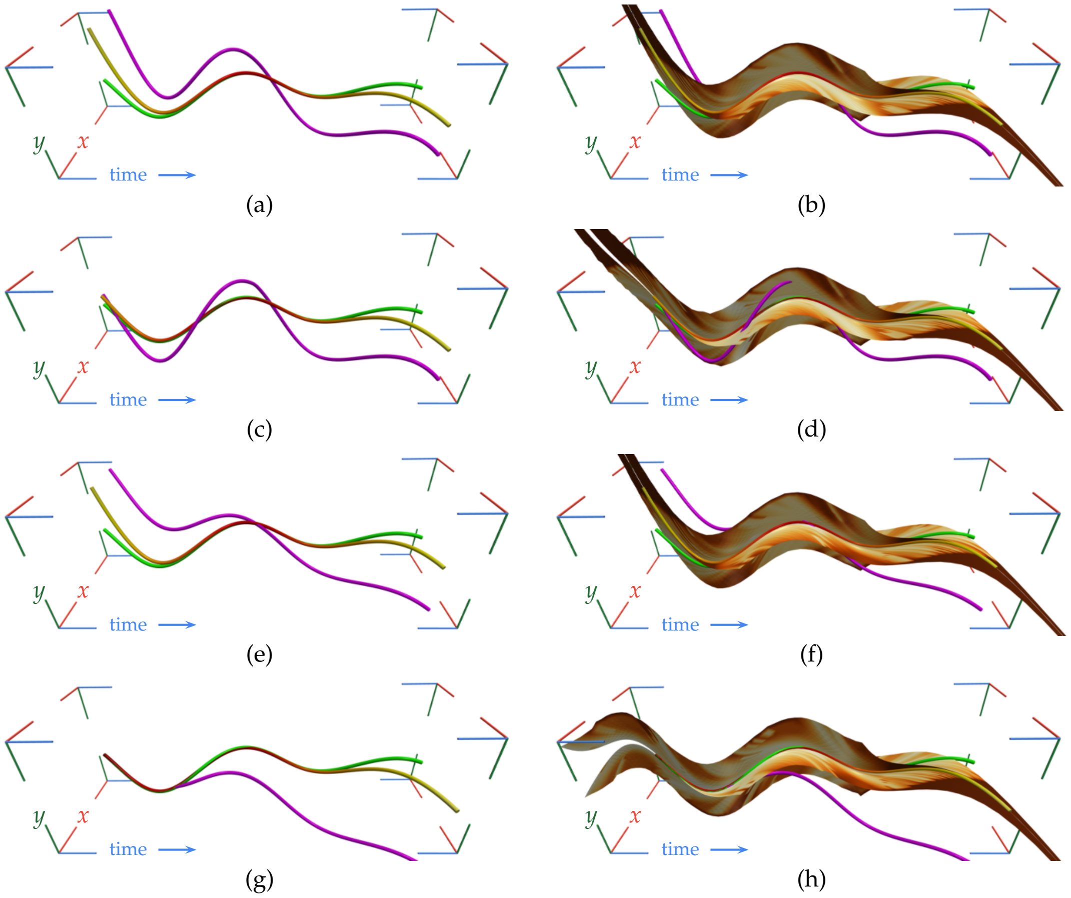


Figure 5.17: Perturbation in the 2D analytic example with ground truth DHT. (a)–(d) Perturbation and refinement of the initial candidate. (e)–(h) Perturbation and refinement of the ground truth. (a)(e) Symmetric perturbation. (c)(g) Linear perturbation. Perturbed candidate shown in magenta, ground truth in green. Refined DHT colored by point-wise distance to ground truth (black-body color map; black: 0, white: 10^{-4} times dataset domain size). Right column: streak manifolds computed from the refined DHTs, colored by pointwise distance to the manifolds of the ground truth DHT (black-body color map; white: 0, black: 10^{-4} times dataset domain size).

nents 3, -3, and a spatial range of 1.0×1.0. We introduce perturbations that linearly vary between zero and $(0.6, 0.6)^\top$. In all cases, our DHT refinement is able to compensate for the perturbations only in directions where sufficient integration time is available. The errors of the streak manifolds are partially corrected by streak integration. Second, we perform the same analysis for the 2D Cylinder Flow (Section 5.6.1), where we use the path of the instantaneous critical point behind the cylinder in the time interval $[0, 2.5]$ as initial candidate, and use the DHT computed from it as ground truth. We compute its Lyapunov exponents as approximately 5.56663, -5.47087, and a spatial range of approximately 0.5×0.02. We impose, relative to the spatial scales, similar per-

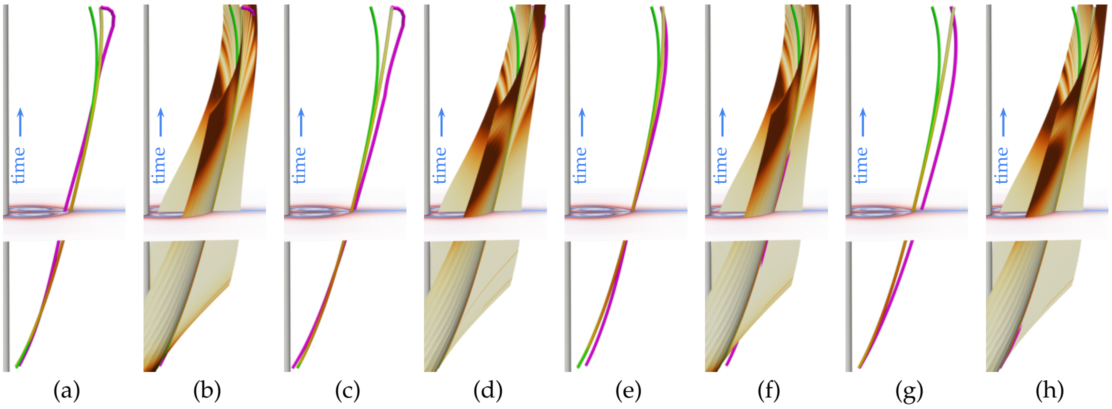
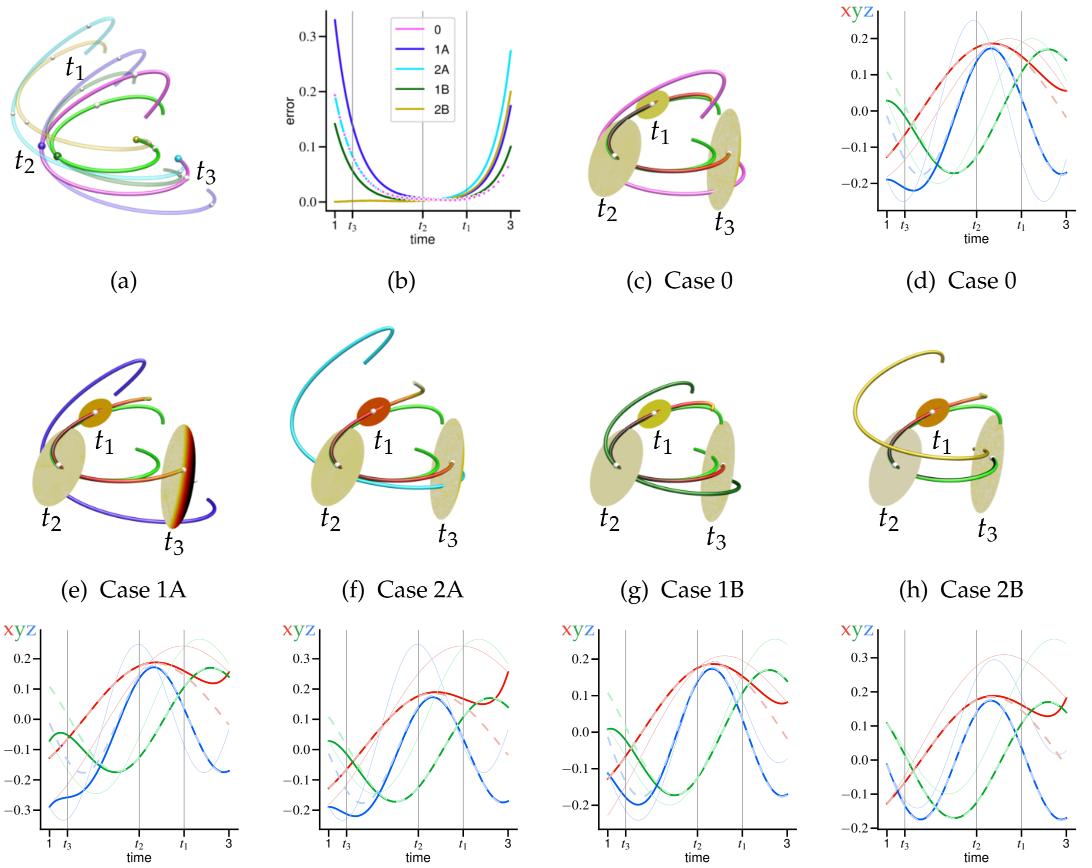


Figure 5.18: First 2.5 s of the 2D Cylinder Flow dataset, analogous perturbation analysis to Figure 5.17, with DHT obtained from unperturbed candidate line used as ground truth. (a)–(d) Perturbed initial candidate. (e)–(h) Perturbed DHT. Same coloring scheme used as in Figure 5.17.

turbations varying between zero and $(0.06, 0.06)^\top$. Similar results as in the analytic case can be observed here (Figure 5.18). However, larger perturbations have caused divergence of the DHT computation in our experiments. While there are theoretical results on this [JSW03], we leave further investigation at numerical datasets for future work.

Perturbation Analysis in 3D. We analyze the stability of the 3D DHT refinement under perturbation of initial candidates using the extension to 3D of the analytic example in Equation 5.17. We fix the parameters $d_1 = 5$, $d_2 = -5$, $d_3 = -4$, $\omega_1 = 2$, $\omega_2 = 3$, $\omega_3 = 4$, $A_1 = A_2 = A_3 = 1$, and sample the analytic field on a regular grid with 101^4 nodes over the domain $[-10, 10]^3 \times [1, 3]$. The path of the saddle-type critical point in the lab frame of reference is taken as initial candidate line. Like in the 2D case, initial candidate (Case A) and ground-truth DHT (Case B) are perturbed symmetrically (Case 1) by $\Delta = 0.1(t-2)$, and asymmetrically (Case 2) by $\Delta = 0.1(t-1)$, identically in all three spatial coordinates. Figure 5.19a shows an overview of the initial candidates and the ground-truth hyperbolic trajectories.

Since the refinement relies on integration along the candidate line, its quality increases with available integration time, i.e., the point-wise error is minimized at the center of the available time interval (Figure 5.19b). By considering the repelling (x, red in Figure 5.19) and attracting (y and z, green and blue in Figure 5.19) directions separately, we see that the repelling direction is refined closer to the ground truth toward the beginning of the time interval, while the attracting directions are refined better toward the end of the time interval (Figures 5.19d and 5.19i to 5.19l). The bends of the refined lines at the ends of the time interval are caused by zero available integration



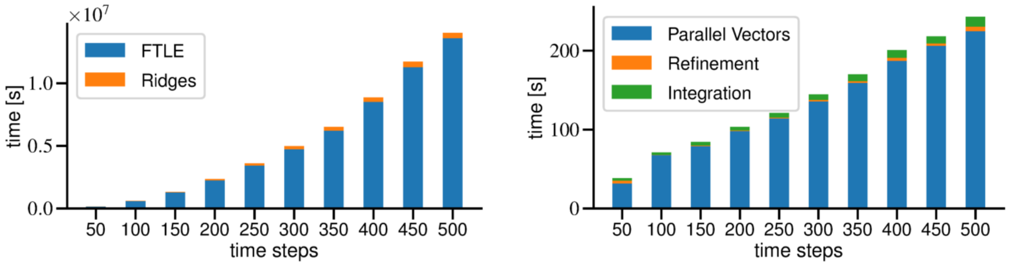
(a) (b) (c) Case 0 (d) Case 0

(e) Case 1A (f) Case 2A (g) Case 1B (h) Case 2B

(i) Case 1A (j) Case 2A (k) Case 1B (l) Case 2B

Figure 5.19: 3D analytic example with ground truth DHT. Perturbation of the 3D initial candidate (A) and the 3D ground-truth DHT (B), symmetrically (1) and asymmetrically (2). Overview shown in (a), with ground truth DHT (green), candidate line (magenta), and the four different perturbation cases (colors as in (b), zero perturbation marked by colored sphere). (c),(e)–(h) Streak manifolds (surfaces, colored by error black to white) computed from the refined HT (colored by error black to white) further compensate errors with increasing advection times. (d),(i)–(l) Plots of the coordinate functions (x red, y green, z blue) of the initial candidate (thin desaturated), the refinement (solid saturated), and the ground truth (dashed). Note that the dashed lines correspond to the green lines, the thin desaturated lines correspond to the single colored lines, and solid lines correspond to lines colored by error in the 3D views (c),(e)–(h).

time for refinement ($t = 1$ for y- and z-coordinates, $t = 3$ for x-coordinate), thus the coordinates of the refined lines are identical to the candidate line there. Only in Case 2B (Figure 5.19l), where the ground truth was perturbed with zero perturbation at the beginning of the time interval, all three coordinates are refined most accurately at the beginning of the time interval as well.



(a)

(b)

Figure 5.20: Performance analysis for the synthetic 3D unsteady saddle dataset. Ridge surface extraction from forward- and backward-time FTLE at each time step for resolution 300^3 of FTLE (a), and our method over the entire dataset (b). Notice time scale difference of about 10^5.

During streak integration, the streak manifolds are attracted toward the corresponding LCS (toward the attracting LCS in forward-time, and toward the repelling LCS in backward-time). This means that the error of the streak manifolds decreases for long integration times, even when started from an erroneous hyperbolic trajectory. To demonstrate this, we compute a backward-time streak manifold seeded at the refined hyperbolic trajectory. The point-wise distance to the streak manifold seeded with the same offset from the ground-truth DHT is shown mapped to color (white low to black high) on the streak manifold at times $t_1 = 2.5$, $t_2 = 2.0$, and $t_3 = 1.2$ in Figures 5.19c and 5.19e to 5.19h. We see that in all cases the error of the streak surface decreases with increasing integration time. The streak surface at time t_3 in Case 1A (Figure 5.19e) exhibits a large error, where it deviates from the ground-truth, possibly caused by the choice of seeding offset.

5.5.5 Performance

We compare the computational costs of our approach with the direct extraction of LCS from FTLE fields (Figure 5.20) at the synthetic 3D unsteady saddle dataset (Section 5.5.1). The direct approach requires the computation of a dense set of pathlines. For comparison, we keep a rather low fixed resolution of 300^3 samples and extract ridge surfaces in the forward and backward FTLE fields for a varying number of time steps (Figure 5.20a). The computational costs depend linearly on both the number of samples and number of integration steps, however the extraction needs to be repeated at each time step, leading to an asymptotically quadratic complexity with respect to the number of time steps. In practice, one would need to increase the resolution of the FTLE fields with increasing advection times in order to accurately resolve the stretching and folding of the LCS, which would result in exponential computational complexity. Our method, on the other hand, has no resolution parameter, but only extracts parallel vectors lines from each original grid cell, which is resolved at 41^3 nodes. Its computational costs also depend linearly on the number of nodes, and thus on the number of time steps (Figure 5.20b). Since the refinement and streak integration is performed over the entire time span of the dataset, this computation yields the LCS for all times. We thus conclude that our method is typically several orders of magnitude faster than a direct extraction, and additionally no numerically challenging evaluation of the flow map or ridge extraction is necessary, similarly as reported by Machado et al. [MBES16] in 2D.

5.5.6 Convergence of FTLE Ridges

A small area of interest (I in Figure 5.24e) of the streak topology computed in the 2D Cylinder Flow (see Section 5.6.1 for a detailed discussion) is shown enlarged in Figure 5.21. At moderate resolution, the ridge locations in the forward and backward FTLE fields cannot be reliably determined (Figure 5.21a). Only with much increased resolution (Figure 5.21c), the five ridges in the backward FTLE field (blue) become apparent. Increasing resolution even further (Figure 5.21d), leads to aliasing artifacts due to the Runge-Kutta 4 integrator with fixed step size used in our implementation. We also note that only one of the five ridge intersections actually belongs to a DHT (see Section 5.4.5), as detected by our streak topology extraction. Except near the seeding locations of the streak manifolds, where numerical integration time is zero, extracted ridges in the FTLE fields approach our streak manifolds for increasing resolutions of FTLE. Figures 5.21e to 5.21h show distances between streak manifolds and ridges.

The same can be observed for the 3D Von Kármán Vortex Street dataset. Figures 5.21i to 5.21l show the convergence of FTLE ridges to our streak manifolds in a small area of the dataset (Figure 5.32c, box). Also note numerical aliasing in the FTLE ridge extraction, which appears due to the same filter parameters being used in each resolution.

5.6 Evaluation on Numerical Datasets

In this section, we evaluate our approach on numerical fluid simulation datasets.

5.6.1 2D Cylinder Flow

We now evaluate our method at a 2D CFD simulation of a flow behind a cylinder. It was computed using the Gerris flow solver [Pop04] and is provided by Günther et

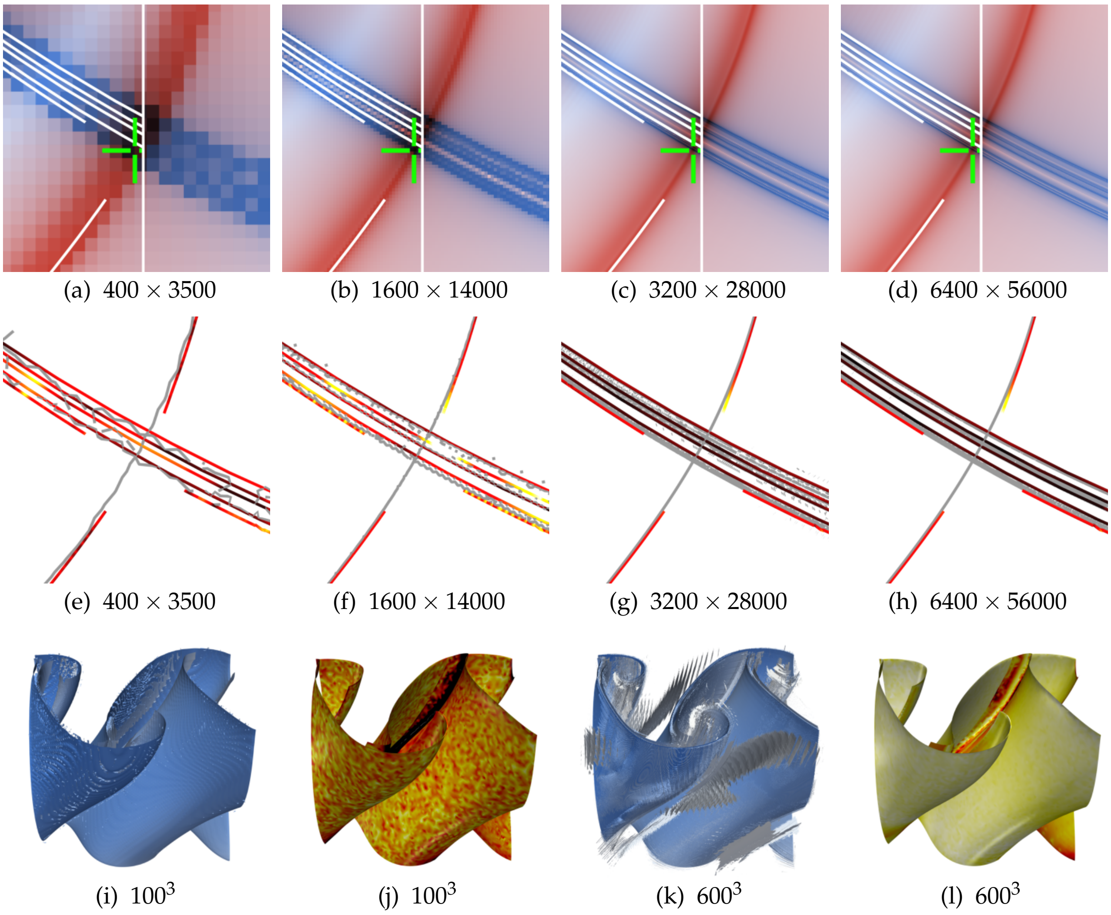
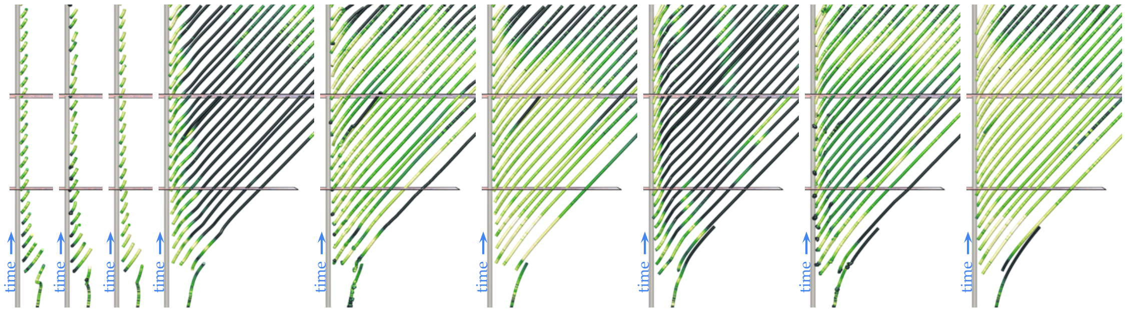


Figure 5.21: Top and middle rows: 2D Cylinder Flow at t=5 s, enlarged region (I) of Figure 5.24e. Comparison of streak manifolds (white lines) with integration time 5 s, and location of DHT (green marker) at t=5 s. Foward (red) and backward (blue) FTLE fields with 5 s advection time, with virtual resolutions 400 × 3500 (a), 1600 × 14000 (b), 3200 × 28000 (c), 6400 × 56000 (d) exhibit ridges that converge to our solution. (e)–(h) Extracted ridges (gray) and distance to nearest ridge (black-body color map; black: zero, white: one cell size). Bottom row: for increasing resolutions of FTLE, ridges (i)(k) converge toward streak manifolds (j)(l) in the 3D Von Kármán Vortex Street (box in Figure 5.24e). Distance to FTLE ridge in white to black (one cell diagonal).

al. [GGT17]. We compute DHTs and the time-dependent vector field topology over the first ten seconds of the dataset. We compare the different extraction methods for obtaining initial candidates, and compare the computation of DHTs with the bifurcation line refinement method by Machado et al. [MSE13]. The results are evaluated using the distance to the nearest intersection of ridges in the FTLE fields. Since extracting ridge lines or ridge surfaces in the space-time domain [BSDW12] requires a prohibitively large FTLE resolution (see Section 5.5.6), we extract local maxima in the product of the two FTLE fields, where we remove critical point pairs with persistence less than 0.5. The



(a) (b) (c) (d) raw (e) [MBES16] (f) ours (g) raw (h) [MBES16] (i) ours

Figure 5.22: Extraction of DHTs in the 2D Cylinder Flow. Initial candidates from lab frame of reference (a), Galilean-invariant frame of reference defined by the feature flow field (d), and displacement-invariant frame of reference [BG20] (g), are corrected toward DHTs using local refinement [MSE13] (b)(e)(h) and with our method (c)(f)(i). Color indicates distance to nearest FTLE ridge intersection (scale shown in Figure 5.23a), with ground truth shown in Figure 5.23e.

topological simplification was performed using TTK [Tie+17]. Using an FTLE resolution of 2000 × 1000, this approach yields reasonably noise-free results (see Figures 5.23b to 5.23d), except where the FTLE ridges are not sharp enough or too close to each other.

Extracting critical points from the lab frame of reference only results in short candidate lines (Figure 5.22a), because the saddles and nodes that are periodically generated behind the cylinder cancel each other out after a short amount of time in this frame of reference. Both our DHT refinement (Figure 5.22c) and the bifurcation line refinement [MSE13] (Figure 5.22b) are only able to make minor corrections, since both methods rely on sufficiently long candidate lines. Computing streak manifolds from these short segments misses most of the repelling LCS, but is already able to obtain large parts of the attracting LCS (Figure 5.23l).

Extracting critical points in the Galilean-invariant frame of reference defined by the feature flow field [MBES16], and in a displayment-invariant frame of reference [BG20] both result in missing line segments due to numerical noise in the area behind the cylinder. Therefore, we obtain initial robust segments by filtering, where $\det \nabla \mathbf{w} > -10$, i.e., τ_h=10, and integrate from their ends along the respective observer motions (Figures 5.23f and 5.23g), as long as integration stays in a hyperbolic region. While in the Galilean-invariant reference frame, integration for some candidate lines leaves the hyperbolic region before the domain boundary is reached, using a displacement-invariant reference frame, the domain boundary is always reached. An exception is the first hyperbolic trajectory, which is created at the beginning of the simulation. It stops existing before t=5 s, since the backward FTLE ridge belonging to its attracting manifold does not intersect with a forward FTLE ridge at this instance of time (II in Figure 5.24e).

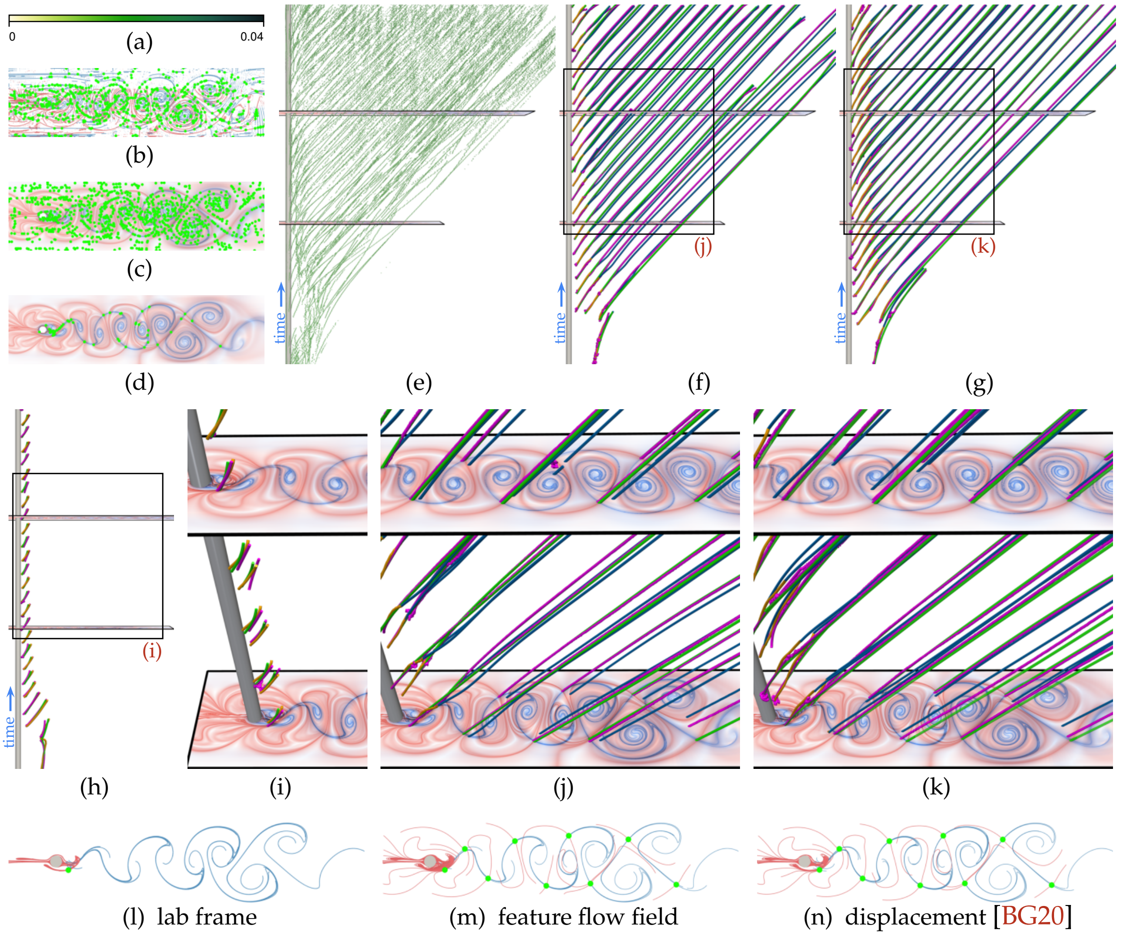
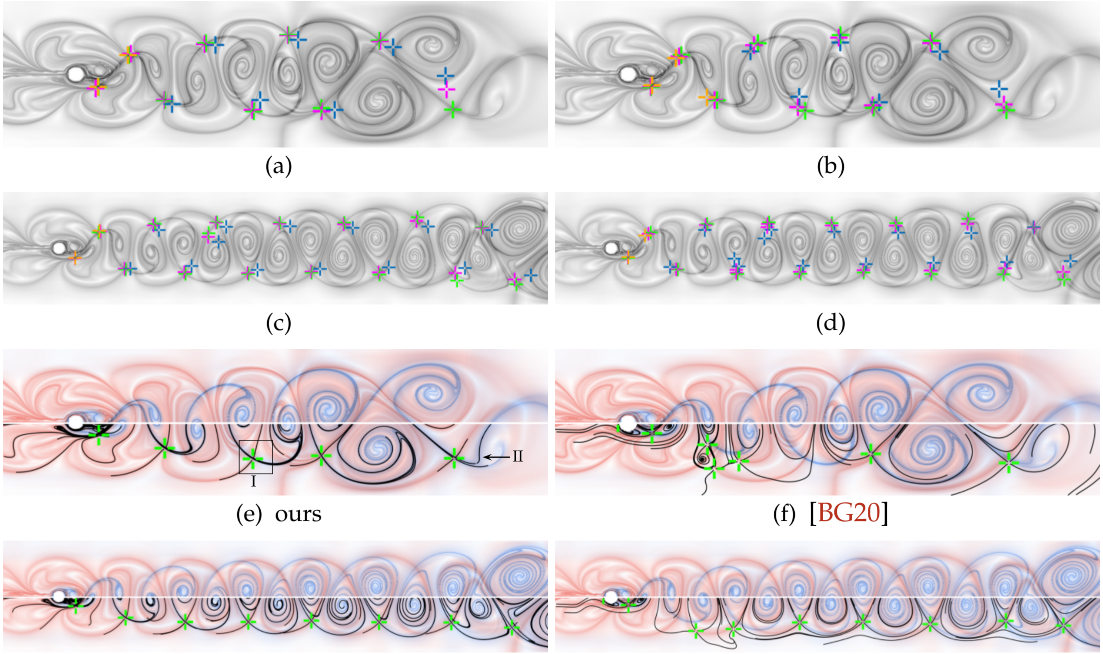


Figure 5.23: Color scale (a) and ground truth (e) used in Figure 5.22, obtained as persistent local maxima in the product of the FTLE fields (d), instead of intersection of ridge lines (b) or ridge surfaces in space-time (c) (at t=5 s). Space-time view ((h): lab frame, (f): feature flow field, (g): displacement-invariant frame of reference) showing initial line segments (orange), extended along observer motion (blue) and refinement (magenta: [MSE13], green: ours), enlarged regions shown in (i)–(k). (l)–(n) Forward (blue) and backward (red) streak manifolds computed from the DHTs (green) at time t=5 s, using the initial candidates from each of the respective approaches.

While the candidate lines in the displacement-invariant frame of reference are more accurate, our DHT refinement reaches similar results in both cases, which very closely follow FTLE ridge intersections (Figures 5.24a–5.24d), except near the right domain boundary, where the ends of the initial candidates are reached. The bifurcation line refinement is very unstable near the cylinder, where the refined lines oscillate. Furthermore, it deviates more from the FTLE ridge intersections near the right domain boundary than our DHT refinement. Streak manifolds computed from the DHTs obtained from both approaches capture most of the attracting and repelling LCS (Figures 5.23m



(a)

(b)

(c)

(d)

(e) ours

(f) [BG20]

(g) ours

(h) [BG20]

Figure 5.24: Refinement (bifurcation lines [MSE13] magenta, and ours green) of initial candidate lines (initial segments yellow, extended by feature flow blue) of critical points in the Galilean-invariant frame of reference defined by the feature flow field, at t=5 s (a) and t=7.5 s (c), and of critical points in the displacement-invariant frame of reference [BG20], at t=5 s (b) and t=7.5 s (d). Streak topology and VFT in the steady frame of reference [BG20] at t=5 s (e)(f) and t=7.5 s (g)(h). (I) Shown enlarged in Figure 5.21. (II) The attracting manifold of the DHT that exists during the first 4 s retains its separating structure also at t=5 s. It does not intersect with a repelling manifold, because the DHT that generated the streak manifold has ceased to exist at this time.

and 5.23n). Since initial candidates obtained using the optimal reference frame [BG20] are longest, they yield larger streak manifolds. Thus, this is the most accurate option, and we propose to use this to obtain candidates for DHT refinement in our streak-based topology. Figures 5.24e–5.24h show a comparison of the VTF in a steady frame of reference as proposed by Baeza Rojo and Günther [BG20] with our method, for the time steps t=5 s and t=7.5 s. While our streak-based topology closely matches ridges in the FTLE fields, critical points in the steady VFT only are accurate in regions, where the LCS are moving at constant speed. Separatrices in the steady VFT only follow FTLE ridges for short amounts of time, if at all. Since streamlines of a steady vector field cannot intersect, intersections of the repelling and attracting LCS, i.e., unsteady saddle connectors, cannot be captured by this concept (see Section 5.4.5 and Figure 5.21).

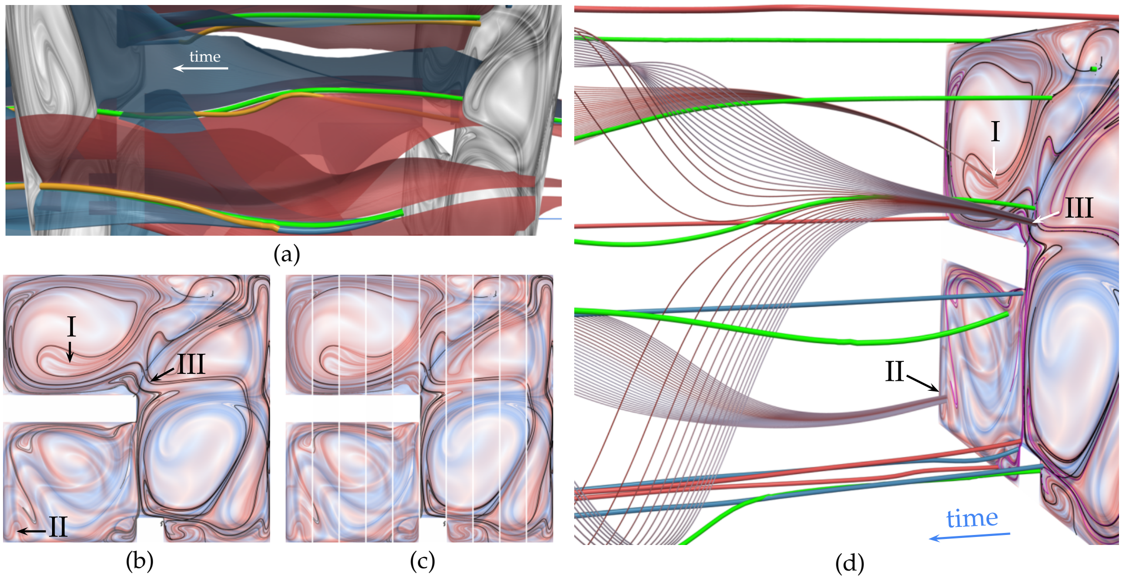
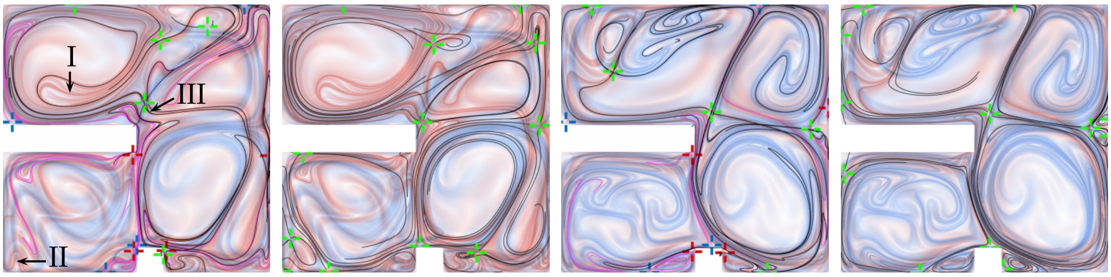


Figure 5.25: (a) Space-time representation of 2D Convective Flow dataset. Paths of saddle-type critical points (orange, extended along observer motion (blue lines)), in a displacement-invariant frame of reference [BG20] over time interval [49 s, 52 s], are corrected toward DHTs (green lines). Streaklines seeded along DHTs in forward (blue surfaces) and backward (red surfaces) time represent streamsurfaces in space-time (with shorter integration times here for clarity) and correspond to LCS (ridges in the FTLE fields, shown here in grayscale). (b)(c) Time step t=50 s (right in (a)) with streakline-based repelling and attracting LCS (black lines, and LCS computed from space-time attachment and separation lines as gray lines (magenta in (b))), FTLE for comparison in slices (c). (d) Pathlines, colored by FTLE, seeded across ridges corresponding to shear (I, II), not covered by the streak topology, do not exhibit strong separation, while ridges contained in the streak topology (black, gray, magenta), are caused by strong separation (III).

5.6.2 2D Convective Flow

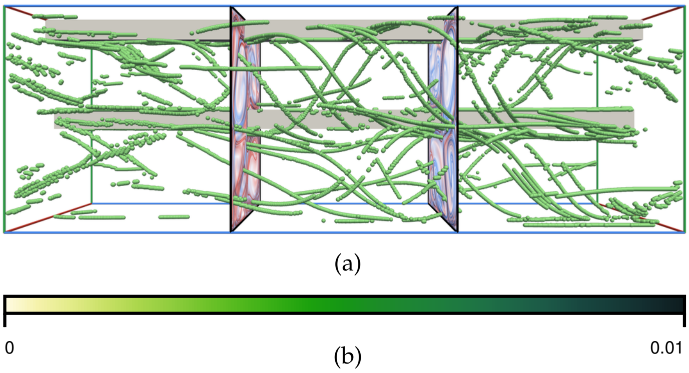
We now consider a CFD simulation of 2D buoyant air flow with two obstacles. This flow is confined to a closed container on a spatial domain of 0.1 m^2 with no-slip boundaries. We consider the time interval $[49\,\mathrm{s}, 52\,\mathrm{s}]$ of this dataset. An overview of its topology in space-time is shown in Figure 5.25a. In this dataset, many FTLE ridges are not captured by our approach. Some of these cases are investigated in Figure 5.25d, where we seed pathlines across some of those missed FTLE ridges. These either correspond to separation induced by shear, or other weak separation, and are thus false-positives in the FTLE field. In Figure 5.26, we compare the streak topology obtained in this way with the steady VFT proposed by Baeza Rojo and Günther [BG20]. Again, neither do the critical points resemble LCS intersections, nor do separatrices follow LCS.

We extract the local maxima in the product of the forward and backward FTLE fields at a resolution of 1000×1000 from 1001 equidistant points in time within the temporal



(a) ours (b) [BG20] (c) ours (d) [BG20]

Figure 5.26: Comparison of the streak topology in the 2D Convective Flow at t=50 s (a) and t=51 s (c), obtained from DHTs (green; black: manifolds) and separation (red) and attachment (blue) lines (magenta: manifolds), and VFT in a displacement-invariant frame of reference [BG20] (green: saddle points, black: separatrices) at the same time slices (b)(d), with FTLE.

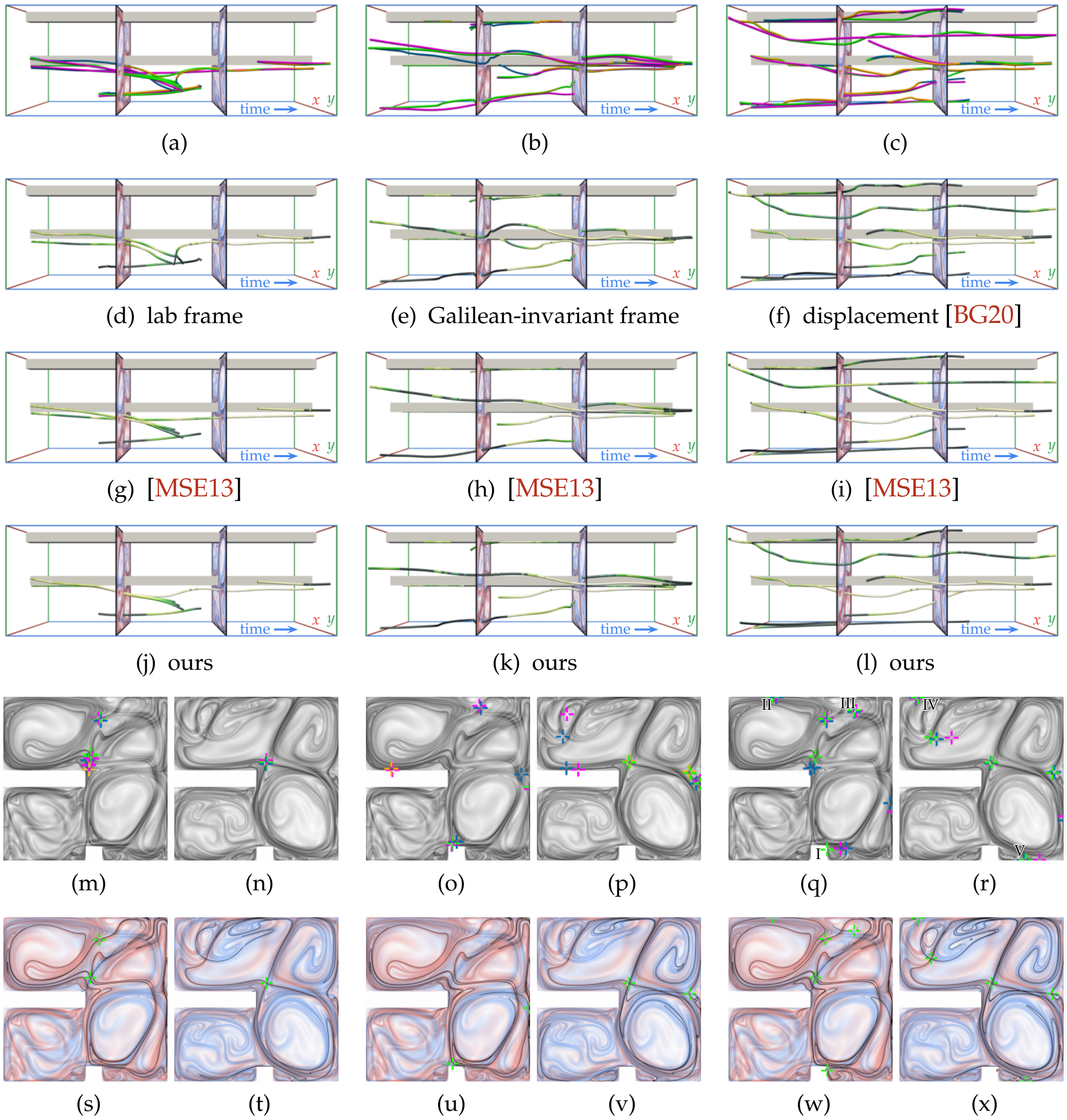


(a)

(b)

Figure 5.27: (a) Persistent local maxima (green) in the product of the forward and backward FTLE fields used as ground truth in the 2D Convective Flow dataset. (b) Color scale for distance to ground truth, used in Figures 5.28d to 5.28l.

domain [49 s, 51 s]. As in Section 5.6.1, we remove critical point pairs with persistence less than 1000, which is approximately 10% (Figure 5.27a). Much of the unsteady topology of the 2D Convective Flow is generated by a slow moving DHT near the center of the domain. An initial candidate can be obtained by tracking critical points in the lab frame (Figure 5.28a). The Galilean-invariant (Figure 5.28b), and displacement-invariant reference frames (Figure 5.28c) are further able to extract candidates for the shorter, weaker DHTs, as well as some of the space-time separation and attachment lines at the no-slip boundaries. While, in these cases, the latter seems to converge to arbitrary pathlines nearby, our DHT refinement reliably reaches either an attracting or repelling LCS, but not necessarily an intersection of both (I–V in Figures 5.28q and 5.28r). Near the boundary, both schemes can leave the spatial domain, since no spatial constraints are imposed by any of the methods. Therefore, we additionally extract space-time separation and attachment lines as proposed by Machado et al. [MBES16] (associated streak manifolds shown in magenta in Figures 5.25 and 5.26). Initial candidates obtained from the displacement-invariant frame [BG20] yield the most accurate results.



(a) (b) (c)

(d) lab frame (e) Galilean-invariant frame (f) displacement [BG20]

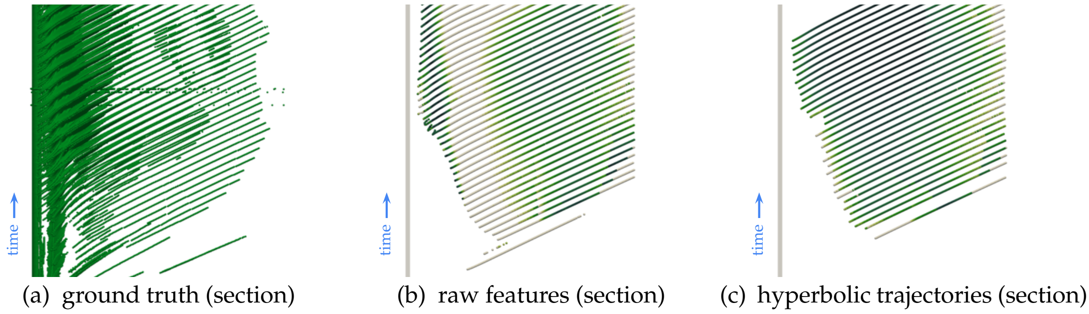
(g) [MSE13] (h) [MSE13] (i) [MSE13]

(j) ours (k) ours (l) ours

(m) (n) (o) (p) (q) (r)

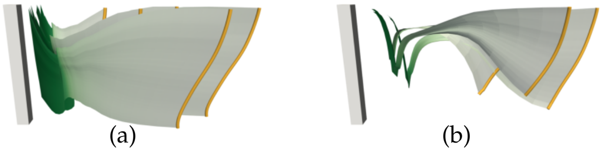
(s) (t) (u) (v) (w) (x)

Figure 5.28: Extraction of DHTs in the 2D Convective Flow. Initial line segments (orange), extended along the respective observer motion (blue), using critical points in the lab frame (a), with time slices (m)(n), in the Galilean-invariant frame of reference defined by the feature flow field (b), with time slices (o)(p), and in the displacement-invariant frame of reference [BG20] (c), with time slices (q)(r). The two time slices shown are at t=50 s and t=51 s, respectively. Initial candidates are corrected toward DHTs using local refinement [MSE13] (magenta) and our refinement (green). (d)–(l) Distance to nearest FTLE ridge intersection for each case (see Figure 5.27 for ground truth and color legend). (s)–(x) Streak manifolds computed from the obtained DHTs in each case (black lines) shown together with forward (red) and backward (blue) FTLE fields. (q)(r),(I–V) At candidate lines that are too close to the boundary or belong to weak DHT, the refinement does not reach FTLE ridge intersections, but are corrected toward nearby LCS instead.



(a) ground truth (section) (b) raw features (section) (c) hyperbolic trajectories (section)

Figure 5.29: The distance of parallel vectors lines (b) in the 3D Von Kármán Vortex Street to FTLE ridge intersections (a) is decreased by refinement toward hyperbolic trajectories (c). Distance in dark green (zero) to white (one cell diagonal). Note that the temporal evolution (time evolves upward) of sections with the plane z=5 is shown (Figure 5.32), i.e., the paths are not trajectories.



(a) (b)

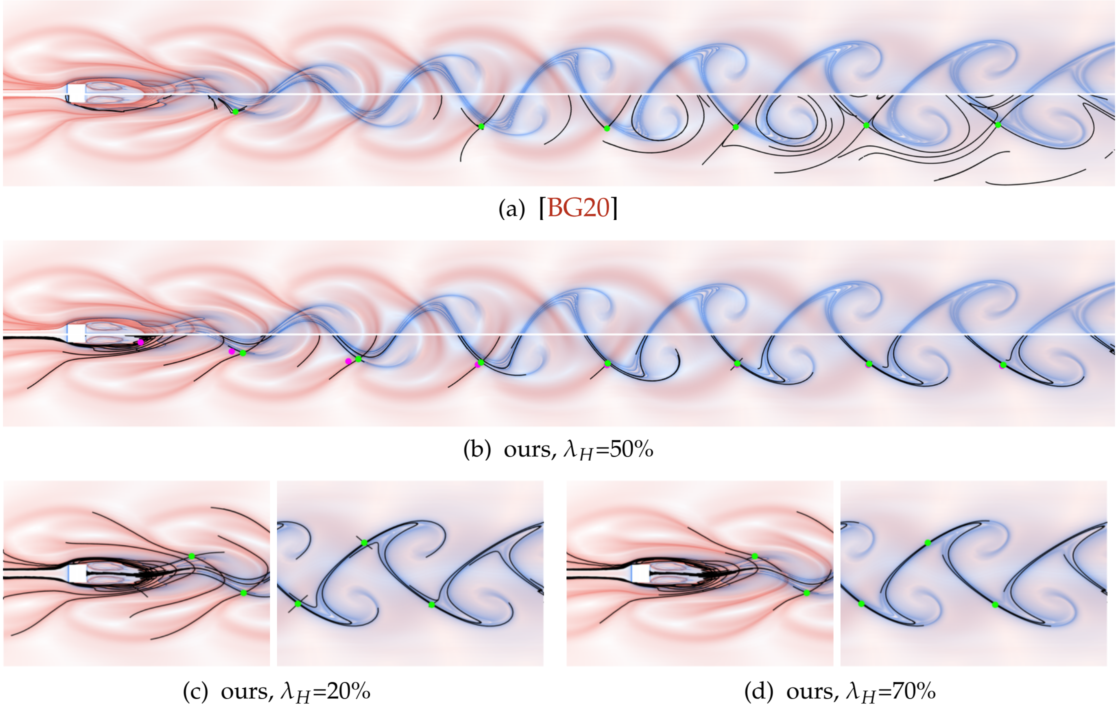
Figure 5.30: Backward-time integration (time in green to white) in the 3D Von Kármán Vortex Street along observer motion $\mathbf{f}(\mathbf{x}, t)$ for Galilean-invariant (a) and displacement-invariant (b) frames of reference started on bifurcation lines (ornage) in $\mathbf{w}(\mathbf{x}, t_0)$ is unstable near the obstacle.

5.6.3 3D Von Kármán Vortex Street

This 3D CFD flow of a von Kármán vortex street forming behind a cuboid obstacle was computed on a domain with extent $[0, 60] \times [0, 10] \times [0, 10]$ m^3 on a $61 \times 41 \times 61$ uniform grid. The dataset contains 801 time steps on the time interval $[0, 0.8]$ s. The inflow velocity varies linearly from 200 m s^{-1} to 250 m s^{-1} with distance from the base.

On the plane $z = 5$, we compute the forward and backward FTLE fields with resolution 2000×1000 at each time step of the dataset. We extract local maxima in the product of the two FTLE fields, where we simplify the scalar topology by filtering critical point pairs with persistence below 2% of the maximum. Due to numerical aliasing, we obtain false-positives near the obstacle (Figure 5.29a), but otherwise a reasonable, computationally feasible approximation of the FTLE ridge intersections. From these, we obtain the distance of the sections of the parallel vectors lines as well as the refined hyperbolic trajectories (Figure 5.29b and 5.29c). We find that the raw solutions are refined well toward FTLE ridge intersections. Note that the lines in Figure 5.29a–5.29c depict the evolution of sections rather than pathlines.

In this dataset, integration along observer motion $\mathbf{f}(\mathbf{x}, t)$ (Figure 5.30) of both the Galilean-invariant and displacement-invariant [BG20] frames of reference is unstable



(a) [BG20]

(b) ours, λ_H=50%

(c) ours, λ_H=20%

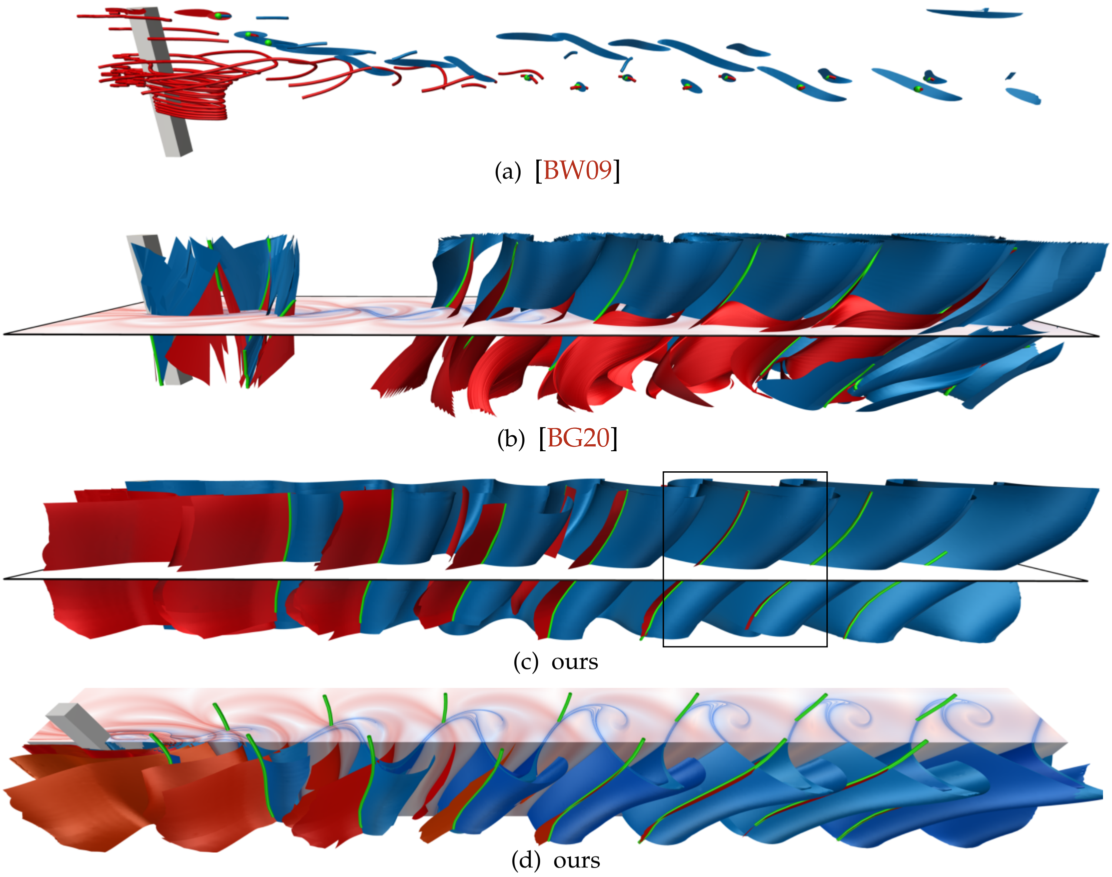
(d) ours, λ_H=70%

Figure 5.31: (a),(b) Sections of the streak topology in the 3D Von Kármán Vortex Street from Figures 5.32b and 5.32c, with manifolds slices shown in black, HTs in green, (b) initial candidates magenta. (b)–(d) Streak manifolds with seeding lengths determined from different percentages of FTLE. We prefer 50%, since its streak manifolds correspond to reasonably sharp FTLE ridges.

near the obstacle, and we were thus unable to recover missing features by integrating along it (cf. 2D case in Section 5.2.1).

Since our dataset exhibits large amounts of numerical noise, we computed derivatives using convolution with derivatives of Gaussians, with standard deviations $\sigma_s = 0.4\,\text{m}$ for spatial derivatives, and $\sigma_t = 0.002\,\text{s}$ for time derivatives. Computation across the entire dataset took 6546 s in total (derivatives and frame of reference: 3197 s, parallel vectors: 2261 s, refinement: 276 s, streak integration: 812 s). Extraction of the ridges in Figure 5.21k, on the other hand, took 253 s, and the corresponding flow map took 2800 s, which only yields a small part of the attracting LCS in a single time step. We note that the backward-time FTLE (blue) yields sharper ridges than forward-time FTLE (red), since backward-time trajectories can get trapped in the recirculation zone behind the obstacle and thus be integrated for longer times before they leave the spatial domain.

Comparing steady VFT in the optimal frame of reference (Figure 5.31a, 3D manifolds shown in Figure 5.32b) to our method (Figures 5.31b, 5.32c and 5.32d), we again see that that steady VFT is not well aligned with the LCS, and this frame of reference exhibits



(a) [BW09]

(b) [BG20]

(c) ours

(d) ours

Figure 5.32: Considering saddle-type critical points in the Galilean-invariant frame **w** of the 3D Von Kármán Vortex Street as initial candidates for hyperbolic trajectories, i.e., a straightforward extension of the 2D method, misses most of the topological structure (a), while vector field topology in a displacement-invariant steady reference frame [BG20] is not aligned with LCS (b). Computed from a Galilean-invariant frame of reference, our method captures large parts of the topology, and is aligned with LCS (c),(d). Slices of (b),(c) are shown in Figures 5.31a and 5.31b.

additional deviations in vertical direction. This computationally more demanding optimal reference frame also does not yield better initial candidates than the Galilean-invariant frame of reference, since there are missing features in a segment behind the obstacle.

In previous work, Branicki and Wiggins [BW09] propose to use 3D critical points in the lab frame of reference as candidates for refinement toward HTs. This dataset, however, only contains critical points on the domain boundary in front of the obstacle, which do not lend themselves for refinement. For comparison, we therefore compute the time-dependent topology from critical points in our Galilean-invariant frame of

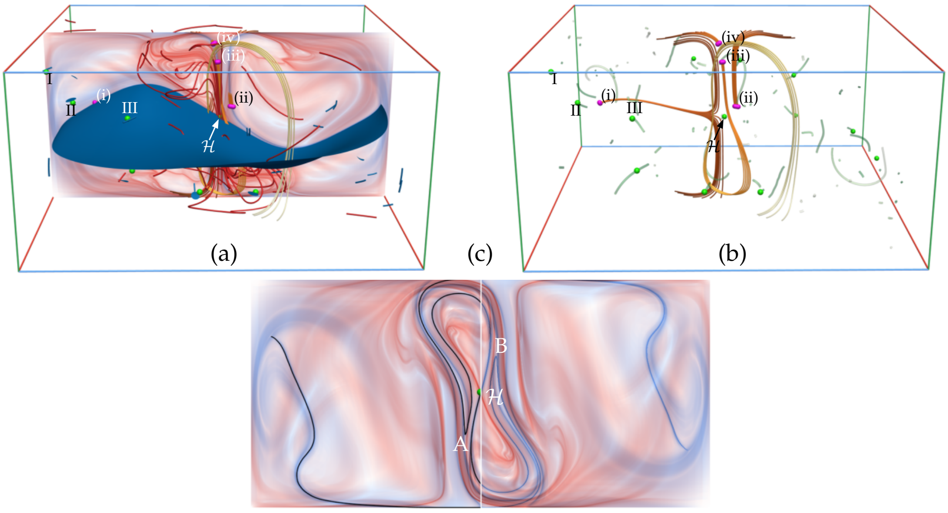


Figure 5.33: 3D Convective Flow dataset on time interval [4 s, 9 s]. (a) At t=5.5 s, the flow is governed by a strongly hyperbolic unsteady spiral saddle $\mathcal{H}$. Streak manifolds of weakly hyperbolic trajectories (I–III) exhibit limited growth. Most FTLE ridges are not captured by our topology because they are shear-induced or correspond to weak separation ((i)–(iv), pathline seeds magenta, integration time yellow to white, pathlines in (b)). (c) Slice of the streak manifold (black) at t=6.5 s, for comparison with FTLE. Note point symmetry, (A) corresponds to (B).

reference (Figure 5.32a), instead of using the lab frame. This only captures small parts of those LCS, where the corresponding bifurcation lines contain a saddle-type critical point, i.e., the LCS are only captured by hyperbolic path surfaces.

5.6.4 3D Convective Flow

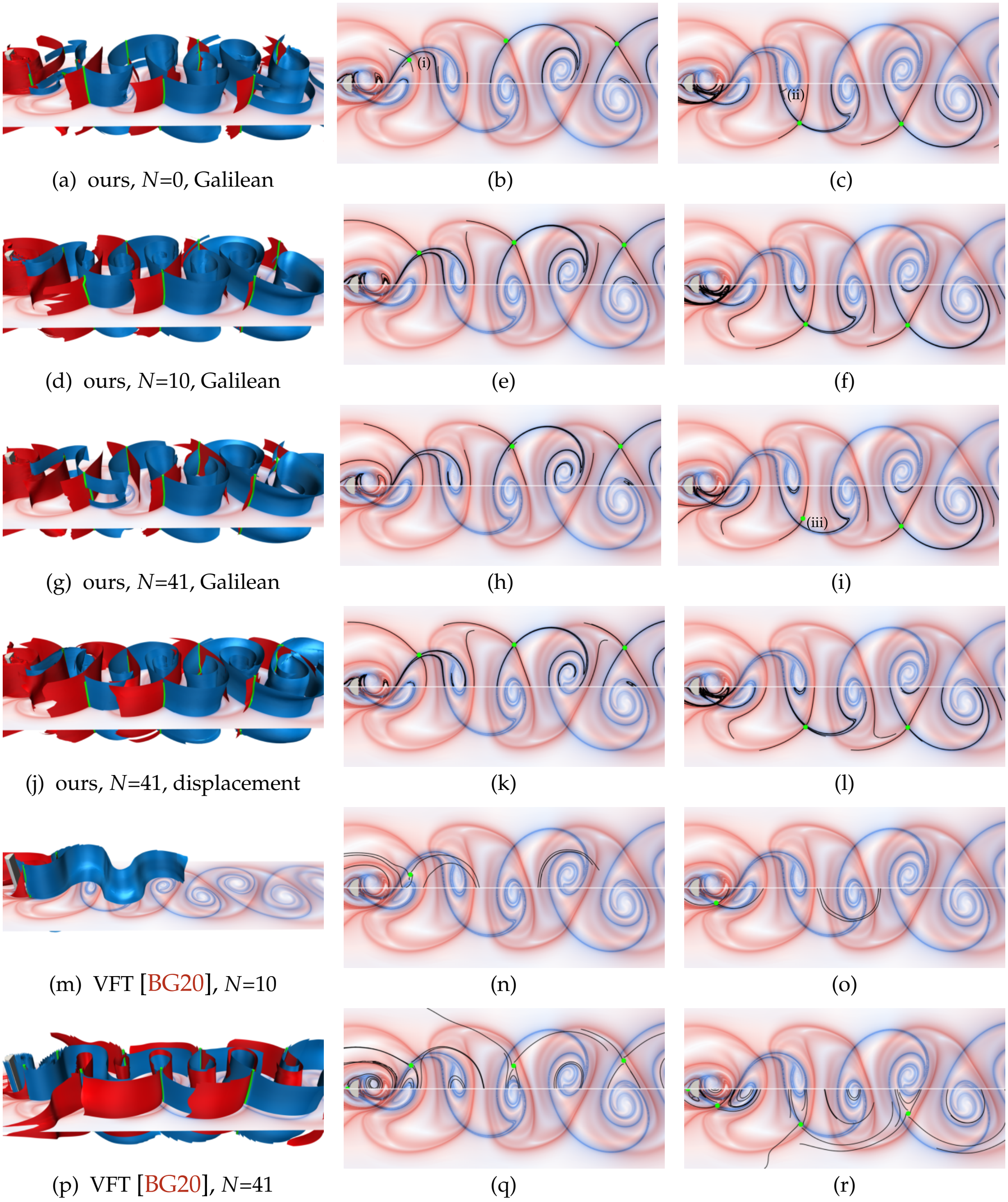
This dataset has extent $[0, 10] \times [0, 5] \times [0, 10]$ m^3 on a $61 \times 31 \times 61$ uniform grid. It contains a CFD simulation of bouyant air flow in a closed container with no-slip boundaries. To ease discussion of this turbulent flow, we select 200 time steps over the time interval $[4, 9]$ s. Computation took 810 s in total (derivatives and frame of reference: 375 s, parallel vectors: 177 s, refinement: <1 s, streak integration: 257 s). The FTLE computed from this dataset exhibits a large amount of thinly folded ridges. However, we found that most of them correspond to weak separation and shear flow (see also Sections 5.4.3 and 5.4.6). Figure 5.33 shows pathline rakes for some of these cases. This makes flow analysis using the FTLE infeasible, since it is unclear how to filter false-positive ridges. In the Galilean-invariant frame of reference, the dataset contains only short lived bifurcation lines, which our method is unable to accurately refine. The separating structure is largely generated by strongly hyperbolic spiral saddle critical points at the domain center, which our approach captures. It also extracts the many weakly hyperbolic saddle-type critical points, which possess streak manifolds of limited growth (I–III in Figure 5.33a), and which could be filtered by a hyperbolicity threshold.

5.6.5 3D Half Cylinder Flow

This flow behind a half cylinder was computed for varying Reynolds numbers using the Gerris flow solver [Pop04] and is provided by Baeza Rojo and Günther [BG20]. The dataset, which was computed on an adaptive grid, was resampled to a uniform grid with dimensions 640 × 240 × 80 × 151 on the space-time domain $[-0.5, 7.5] \times [-1.5, 1.5] \times [-0.5, 0.5] \times [0, 15]$. We use two members of the ensemble with Reynold numbers $Re = 160$ and $Re = 320$ to analyze the behavior of our method at varying degree of turbulence. Seeding lengths for streak manifolds were determined using the method described in Section 5.2.4, using an FTLE percentage of 50%.

At $Re = 160$, the flow exhibits low turbulence. We employ our proposed Galilean-invariant frame of reference for varying sizes of neighborhoods (Equation B.2). Direct inversion of the Jacobian to compute the feature flow field corresponds to an infinitesimally small neighborhood N=0. Due to numerical noise, some initial candidates are missed, which leads to low quality in our extracted hyperbolic path surfaces (Figures 5.34a to 5.34c). A too large neighborhood of radius N=41 nodes, on the other hand, leads to overly smoothing and thus false negatives (Figures 5.34g to 5.34i). Our proposed rather small neighborhood of N=10 nodes, provides the best results (Figures 5.34d to 5.34f). For comparison, VFT in the steady, displacement-invariant frame of reference [BG20] is not well aligned with LCS for both neighborhood sizes (the authors propose to use N=41 in their work). However, these features provide good initial candidates for our refinement, which leads to a more extendedly extracted topology (Figures 5.34j to 5.34l). Note that this comes, however, at additional computational costs at about 930 min, as well as a rather large memory requirement of about 360 GB of RAM to store the summed area table as well as the dataset and its derivatives, compared to the Galilean-invariant optimization, which, on the other hand, took about 150 min and required about 50 GB of RAM.

At $Re = 320$, where the flow is more turbulent, all frames of reference fail to provide suitable initial candidates. We employ the same methods as in the previous case in Figure 5.35. Since all frame of references to not provide long enough initial candidates for our refinement, the obtained hyperbolic path surfaces have large distances to the actual LCS intersections. The streak manifolds, however, are still attracted toward the LCS during integration, and are thus better aligned with them.



(a) ours, *N*=0, Galilean (b) (c)

(d) ours, *N*=10, Galilean (e) (f)

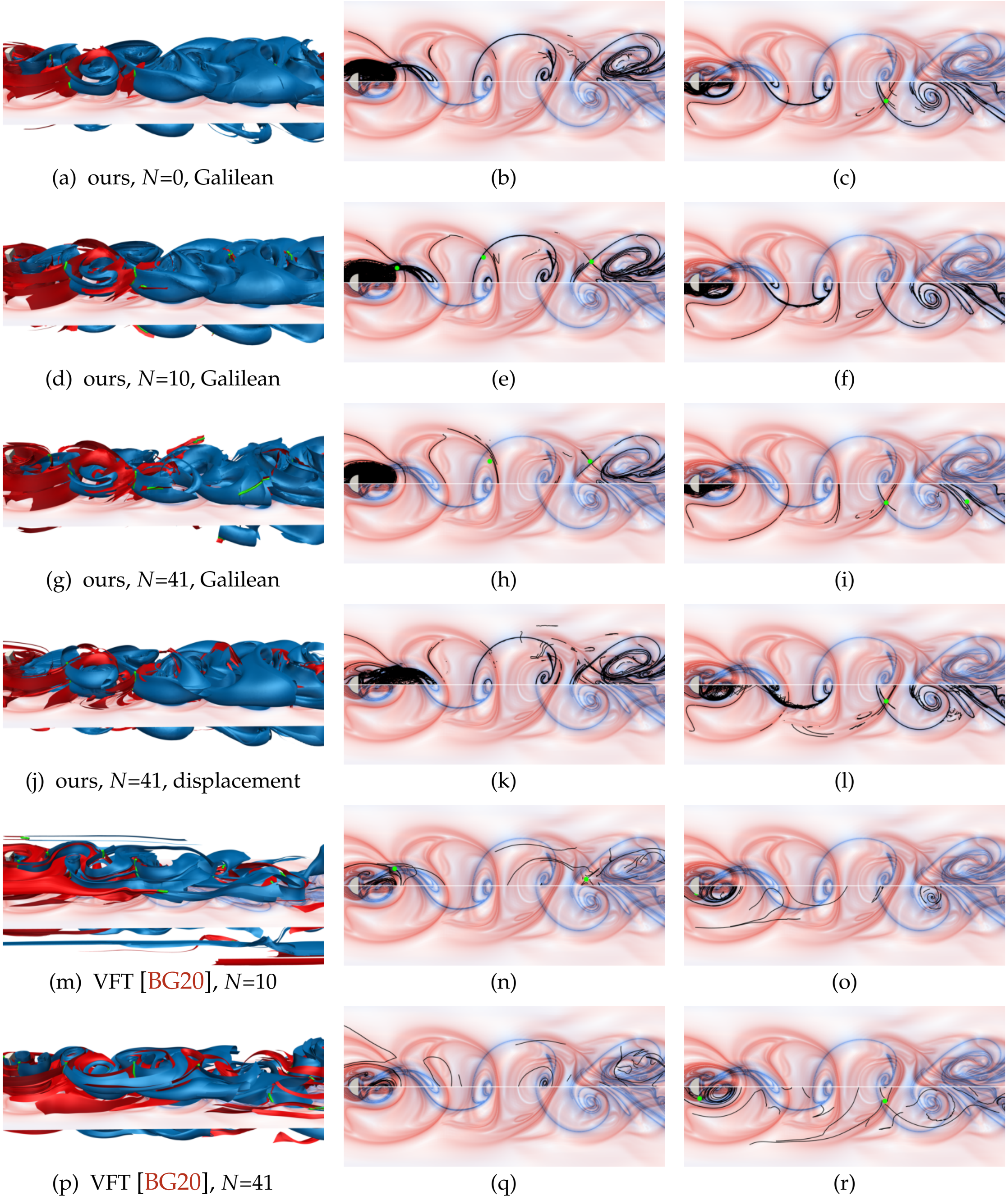
(g) ours, *N*=41, Galilean (h) (i)

(j) ours, *N*=41, displacement (k) (l)

(m) VFT [BG20], *N*=10 (n) (o)

(p) VFT [BG20], *N*=41 (q) (r)

Figure 5.34: 3D Half Cylinder Flow with modest turbulence at *Re*=160. Black lines in mid and right column represent sections of manifolds at shown slice in left column. Direct inversion of the Jacobian, (a)–(c), suffers from numerical noise, and yields worse initial candidates (i) than using a neighborhood of *N* nodes. A too large neighborhood *N*=41, (g)–(i), has an overly smoothing effect (iii) and results in false negatives. We thus prefer *N*=10, (d)–(f). While VFT in the displacement-invariant frame of reference [BG20] is not well aligned with the LCS, (m)–(r), it provides good initial candidates for our method, (j)–(l), at increased computational costs.



(a) ours, N=0, Galilean (b) (c)

(d) ours, N=10, Galilean (e) (f)

(g) ours, N=41, Galilean (h) (i)

(j) ours, N=41, displacement (k) (l)

(m) VFT [BG20], N=10 (n) (o)

(p) VFT [BG20], N=41 (q) (r)

Figure 5.35: 3D Half Cylinder Flow with higher turbulence at Re=320, with same methods applied as in Figure 5.34. Both the Galilean-invariant and displacement-invariant [BG20] frames of reference fail to extract suitable candidate lines in this turbulent dataset. Our method with N=10 again performs comparatively best, but also still misses large parts of the LCS. Nevertheless, streak integration is attracted by the respective LCS for long enough integration times, and is thus able to partially correct for the large initial errors.

5.7 Discussion

Our approach is based on the refinement of initial candidates toward hyperbolic trajectories. This comes with two major limitations. Since the refinement is integration-based, it requires sufficiently long initial candidates, depending on the dataset. The sharpness of ridges in the FTLE fields computed within the time interval of an initial candidate can give an indication whether the candidate is sufficiently long. Domain boundaries can additionally limit the available integration time in cases, where integration leaves the domain boundaries.

Secondly, candidate extraction is based on tracking critical points and bifurcation lines in a suitable frame of reference. The extraction of bifurcation lines within a single time step can itself exhibit false negatives, especially for feature lines with high curvature [MSE13]. Finding an optimal frame of reference is also challenging in the presence of turbulence (see Section 5.6.5), which can further lead to errors in the feature tracking. A good frame of reference for these cases has yet to be found and would be required for ensuring a robust extraction of the time-dependent topology in turbulent flow.

Conceptually, our approach is based on the notion of exponential dichotomy, i.e., on the asymptotic behavior of trajectories. Over a finite time interval, this necessarily leads to a lack of uniqueness, which means that our extracted features heavily depend on the quality of initial candidates. Interpreting this non-uniqueness as, say, uncertainty, could be explored in future work to obtain a fuzzy time-dependent topology, which more accurately incorporates the limitations imposed by the finite time interval of the dataset. Another conceptual limitation is the instantaneous hyperbolicity condition, which we impose on our hyperbolic trajectories. We found this to be necessary for a numerically stable refinement, but it is not necessary for the definition due to Ide et al. [ISW02]. Dropping this requirement would include trajectories that undergo transitions between hyperbolic and non-hyperbolic behavior [BW10], but would also pose the problem of defining, at which instance of time a trajectories stops being hyperbolic.

6 Conclusion

In this thesis, we presented several approaches to the visualization of higher-dimensional and time-dependent vector fields. This chapter briefly summarizes our contributions, and discusses their implications as well as possible future work.

The first part of the thesis focused on local feature extraction. To this end, we introduced the dependent vectors operator in Chapter 3, which generalizes the parallel vectors operator due to Peikert and Roth [PR99] to arbitrary dimension. We presented a generic algorithm to extract the resulting k-dimensional manifolds of locations where k+1 fields of n-dimensional vectors are linearly dependent. We demonstrated its applicability on the definition and extraction of higher-dimensional vortex core manifolds, bifurcation manifolds, as well as ridge manifolds. Furthermore, we showed that several recent feature extraction approaches with underlying higher-dimensional problems can also be easily formulated using the dependent vectors operator.

Since the number k of vector fields defines the dimensionality of the solution manifolds, our dependent vectors operator is particularly useful for visualization using projections, as lower dimensionality of the geometry leads to less occlusion and self-intersection in its projections. In future work, further types of features can be readily explored using our technique, such as higher-order [RP98] generalizations of vortex core manifolds. The dependent vectors operator itself could also be generalized further, for example, by requiring that a subset of k out of m vector fields are linearly dependent, such as in the context of eigenvector fields of second-order tensor fields [ORT18b]. Our proposed algorithm provides no guarantees that the obtained geometry is manifold. This could be improved by incorporating tensor field topology [STS09] or feature flow fields [The+05; TS03] for a topologically correct, but possibly more expensive, extraction.

In the second part of this thesis, we focused on vector field topology as a global feature. The topology of steady 4D vector fields was treated in Chapter 4, where we classified the types of critical points and periodic orbits in 4D vector fields, and presented an approach for the interactive visualization of the 4D scenes resulting from extracting their invariant manifolds. For the latter, we exploited degeneracies in the projections from 4D to 3D space to develop a manifold-based exploration technique.

We further presented two strategies for avoiding projection-induced occlusion. The first is a modified projection scheme that avoids self-intersections within a user-defined radius around the 4D camera location, and the second one involves 4D distance fields between the invariant manifolds. We demonstrated the utility of our approach using vector fields that showcase properties of 4D vector field topology, and indicated that saddle connectors can be either lines or surfaces in 4D vector fields.

We found that projections of one- and two-dimensional manifolds in 4D space are easier to interpret, since there is less occlusion and self-intersections. Our modified projection scheme mainly deals with occlusion caused by projection of the three-dimensional manifolds, which tend to still be three-dimensional after projection. Those projection parameters that cause such projections to degenerate to lower-dimensional manifolds are precisely what we exploited to make interactive exploration feasible. In future work, saddle connectors [TWHS03] could be extracted from 4D vector fields to obtain a lower-dimensional representation of the topological skeleton, as well as topological structures defined on obstacles and domain boundaries, such as boundary switch manifolds [WTHS04] as well as attachment and separation manifolds [Ken98]. Since in n-dimensional vector fields, these structures are (n-2)-dimensional, they could also be considered for the visualization of 5D steady vector fields. We expect that the interactive exploration and projection schemes, which we employed in our experiments, need to be tailored to specific use cases for the visualization of more complex geometry in four- and higher-dimensional space. Furthermore, many problems involve different scales among the dimensions, such as the phase space of inertial systems, which is spanned by position and momentum. Such problems often already have canonical projections into the corresponding position- and velocity-subspaces [Sag+17].

In Chapter 5, we presented an extraction method for the topology of 2D and 3D time-dependent vector fields. Our approach relies on local extraction of candidate lines, which are refined toward hyperbolic trajectories. To this end, we combined methods for computing optimal frames of reference [BG20; GT20] with work on computing distinguished hyperbolic trajectories [BW09; ISW02]. In the 2D case, we showed that our approach is faster and more reliable than the previous method [MBES16] based on space-time bifurcation lines [MSE13]. We extended our method to the 3D case, where we extracted hyperbolic path surfaces from 4D candidate surfaces. While, conceptually, our 4D candidate surfaces can be defined and extracted using our dependent vectors operator, its direct extraction turned out to be unreliable in numerical datasets, such that we employed a tracking approach tailored to the time-dependent problem instead. We demonstrated that, unlike existing straightforward extensions of the 2D method [BW09], our approach is consistent with the LCS indicated by FTLE ridges, while being typically orders of magnitude faster and more accurate than FTLE-based approaches. Both 2D and 3D approaches were evaluated on datasets from computational fluid dynamics, using the FTLE as ground truth, where we further showed that our approach is more consistent than VFT in a steady frame of reference [BG20].

Since our method results in a generative geometric representation of the LCS, it lends itself for further numerical analysis. We have already exemplified the extraction of an unsteady equivalent of saddle connectors between hyperbolic trajectories, which are in analogy to saddle connectors between critical points in 3D and 4D steady vector fields. Since it is mainly these connections that form closed regions in the spatial domain of similar flow behavior [MW98], their efficient extraction and visualization could be explored in future work. In complex flows, involving many hyperbolic trajectories of different strengths and time spans, visual analytics approaches could be employed for their analysis. The system proposed by Sagristà et al. [SJS20] could possibly be adapted for this purpose by replacing FTLE ridges with our streak-based topology. As we have shown in Section 5.6.5, our employed heuristic for finding initial candidates of hyperbolic trajectories and path surfaces is numerically unstable in regions of turbulent flow. In future work, other types of reference frames could be investigated, which are tailored to the problem of extracting bifurcation lines in the steady frames. For this, the minimization of the temporal derivative of velocity [BG20; GT20] would possibly need to be replaced by a condition that directly incorporates the parallel vectors operator.

This thesis was largely motivated by a lack of visualization literature extracting features directly from the higher-dimensional spaces that arise from many applications. Much of this work is limited by the curse of dimensionality, which currently limits real-world datasets to a dimensionality as low as six. However, with the rising availability of high-performance computing and in-situ methods, processing such large datasets becomes more and more feasible. Our thesis thus serves as an initial step toward an integrated treatment of feature extraction and visualization for higher-dimensional fields.

Appendix

A Structured Grids in Arbitrary Dimensions

Throughout this thesis, especially for our dependent vectors operator (Chapter 3), we used a generic implementation of a regular grid in nD space. By employing respective coordinate transformations, such an implementation can be extended to any type of structured grid (Section 2.3). For simplicity, we outline an implementation of n-dimensional Cartesian grids.

We define an n-dimensional Cartesian grid by an origin $\mathbf{o} \in \mathbb{R}^n$, and dimensions $\mathbf{d} \in \mathbb{Z}^n$. The grid is then given by the set of nodes

$$\mathbf{n}[\boldsymbol{\iota}] = \mathbf{n}[\iota_1, \dots, \iota_n] = \mathbf{o} + (\iota_1, \dots, \iota_n)^\top, \tag{A.1}$$

for each index vector $\boldsymbol{\iota} \in \mathbb{Z}^n$, where $0 \leq \iota_i < d_i$ for $i = 1, \dots, n$.

Cells. The cells of the grid are described by the translated hypercubes $\mathbf{n}[\boldsymbol{\iota}] + [0,1]^n$, where $0 \leq \iota_i < d_i - 1$. This places the node with index $\boldsymbol{\iota}$ at the "lower left" corner of the cell. The mapping of a point $\mathbf{x} \in \mathbb{R}^n$ to cell-local coordinates is given by

$$\boldsymbol{\iota} = \lfloor \mathbf{x} - \mathbf{o} \rfloor, \quad \boldsymbol{\zeta} = \mathbf{x} - (\mathbf{o} + \boldsymbol{\iota}). \tag{A.2}$$

The point $\mathbf{x}$ lies outside of the grid, if $\boldsymbol{\iota}$ is outside of the domain, i.e., there is an i such that $\iota_i < 0$ or $\iota_i \geq d_i$, or $\boldsymbol{\zeta} \notin [0,1]^n$.

Interpolation. Multilinear interpolation straightforwardly extents to arbitrary dimensions. Consider cell-local coordinates $\boldsymbol{\zeta}$ belonging to a point $\mathbf{x}$ and cell index $\boldsymbol{\iota}$ as above. For each node on the unit cube, $\boldsymbol{\kappa} \in \{0,1\}^n$, we define an interpolation weight $\omega(\boldsymbol{\kappa})$ as

$$\omega(\boldsymbol{\kappa}) = \prod_{\{i|\kappa_i=1\}} \zeta_i \prod_{\{i|\kappa_i=0\}} (1 - \zeta_i). \tag{A.3}$$

Given data $\mathfrak{f}[\boldsymbol{\iota}]$ on the grid nodes, we obtain the interpolated value by the weighted sum

$$f(\mathbf{x}) = \sum_{\boldsymbol{\kappa} \in \{0,1\}^n} \omega(\boldsymbol{\kappa}) \mathfrak{f}[\boldsymbol{\iota} + \boldsymbol{\kappa}]. \tag{A.4}$$

Cell faces. The type of a cell face of dimension k, with $0 \leq k \leq n$, is defined by the coordinate axes which span the k-dimensional parallelotope. Thus, there are $\binom{n}{k}$ different types defined by the choice of a k-subset of the n basis vectors $\mathbf{e}_1, \dots, \mathbf{e}_n$. We represent them as those n-tuples $\boldsymbol{\gamma} \in \{0,1\}^n$ that have exactly k entries 1 and n-k entries 0. A specific cell face is then represented by an index $\boldsymbol{\iota}$ as

$$\boldsymbol{\iota} + [0, \gamma_1] \times \cdots \times [0, \gamma_n]. \tag{A.5}$$

Here, only those indices $\boldsymbol{\iota}$ represent cell faces of the grid that satisfy

$$\begin{aligned} \iota_i &\geq 0, & \text{for} \quad i &= 1, \dots, n, & \text{(A.6)} \\ \iota_i &< d_i, & \text{if} \quad \gamma_i &= 0, & \text{(A.7)} \\ \iota_i &< d_i - 1, & \text{if} \quad \gamma_i &= 1. & \text{(A.8)} \end{aligned}$$

Incident faces. Given a k-face F of type $\boldsymbol{\gamma}$ with index $\boldsymbol{\iota}$, we seek all l-faces ($l > k$) that have F as face. First, we enumerate the possible types $\boldsymbol{\beta}$ of incident faces. These are all types of l-faces, such that

$$\gamma_i = 1 \Rightarrow \beta_i = 1, \quad \text{for} \quad i = 1, \dots, n. \tag{A.9}$$

For each such type $\boldsymbol{\beta}$, there are $l - k$ entries, such that $\beta_i = 1$ and $\gamma_i = 0$. The set $\mathcal{I}$ of indices of faces with type $\boldsymbol{\beta}$ that have F as face is then given by

$$\mathcal{I} = \boldsymbol{\iota} + b_1 \times \cdots \times b_n, \tag{A.10}$$

$$b_i = \begin{cases} \{0,1\}, & \text{if } \beta_i = 1 \text{ and } \gamma_i = 0, \\ \{0\}, & \text{else,} \end{cases} \tag{A.11}$$

where we exclude those indices that do not satisfy Equations A.6 to A.8.

Faces of faces. Given a k-face F of type $\boldsymbol{\gamma}$ with index $\boldsymbol{\iota}$, we seek all j-faces ($j < l$) that are a face of F. This is similar to the enumeration of incident faces, but with the roles reversed. The possible types $\boldsymbol{\nu}$ of such j-faces are those that satisfy

$$\nu_i = 1 \Rightarrow \gamma_i = 1, \quad \text{for} \quad i = 1, \ldots, n. \tag{A.12}$$

For each such type $\boldsymbol{\nu}$, there are $k - j$ entries, such that $\gamma_i = 1$ and $\nu_i = 0$. The set $\mathcal{I}$ of indices of faces with type $\boldsymbol{\nu}$ that are a face of F is given by

$$\mathcal{I} = \boldsymbol{\iota} - b_1 \times \cdots \times b_n, \tag{A.13}$$

$$b_i = \begin{cases} \{0, 1\}, & \text{if } \gamma_i = 1 \text{ and } \nu_i = 0, \\ \{0\}, & \text{else,} \end{cases} \tag{A.14}$$

where we, again, exclude those indices that do not satisfy Equations A.6 to A.8.

Combinatorics. A key ingredient in above formulas are combinatorial algorithms. These can be found, for example, in the book by Knuth [Knu11].

Equations A.4, A.10 and A.13 require to enumerate all sets $\{0, 1\}^n$. These are in 1:1 correspondence to all n-bit binary numbers that can be obtained by counting from 0 to $2^n - 1$. The sets are recovered by checking if each bit is set.

All types of k-faces can be obtained by enumerating all combinations of tuples, where k entries are 1. This can be implemented using a next lexicographic permutation algorithm. Starting from the smallest one, $(0, \ldots, 0, 1, \ldots, 1)$, the next permutation is computed until the largest, $(1, \ldots, 1, 0, \ldots, 0)$, is reached.

In practice, we implemented all involved sets of the form $\{0, 1\}^n$ and their combinatorics using bit arithmetic. We further used C++ template programming with the dimension n as template parameter, which allowed the compiler to unroll all involved loops over n, and provided similar performance as traditional implementations with hard-coded dimension.

B Implementation Details

This chapter provides further details for our time-dependent vector field topology extraction method (Chapter 5).

B.1 Reference Frame Computation

Computation of the displacement-invariant frame of reference was performed using the prototype provided by Baeza Rojo and Günther [BG20] in the 2D case, which we based our implementation for 3D flows on.

For computing a Galilean-invariant frame of reference in the 3D case, we use the method by Günther and Theisel [GT20]. The least-squares problem

$$\int_{\mathbf{x}\in\mathcal{U}} \|\nabla\mathbf{u}(\mathbf{x},t)\mathbf{f}(\mathbf{x},t) + \mathbf{u}_t(\mathbf{x},t)\|^2 \to \min \tag{B.1}$$

is solved for $\mathbf{f}(\mathbf{x},t)$ in its discretized form on a uniform grid. At grid node (i,j,k), we sum over the discrete neighborhood with a ten nodes radius,

$$\mathcal{U} = \{i-10,\dots,i+10\} \times \{j-10,\dots,j+10\} \times \{k-10,\dots,k+10\}, \tag{B.2}$$

$$\mathbf{A} = \sum_{\mathbf{x}\in\mathcal{U}} \nabla\mathbf{u}(\mathbf{x},t)^\top \nabla\mathbf{u}(\mathbf{x},t), \tag{B.3}$$

$$\mathbf{b} = \sum_{\mathbf{x}\in\mathcal{U}} \nabla\mathbf{u}(\mathbf{x},t)^\top \mathbf{u}_t(\mathbf{x},t). \tag{B.4}$$

Nodes outside of the grid are skipped, i.e., are assumed to have zero values in these equations. Finally, the linear system $\mathbf{A}\mathbf{f} = -\mathbf{b}$ is solved using a Householder QR decomposition with full pivoting. Like in the original work [GT20], the summation is performed over the entire grid using a three-dimensional summed area table for better computational performance.

B.2 Computation of Fundamental Solution Matrix

The singular value decomposition,

$$\mathbf{X}_{t_0}(t) = \mathbf{B}_{t_0}(t) e^{\mathbf{\Sigma}_{t_0}(t)} \mathbf{R}_{t_0}(t)^\top, \tag{B.5}$$

of the fundamental solution matrix $\mathbf{X}_{t_0}(t)$,

$$\frac{d}{dt}\mathbf{X}_{t_0}(t) = \mathbf{J}(t)\mathbf{X}_{t_0}(t), \quad \mathbf{X}(t_0) = \mathbb{I}, \tag{B.6}$$

is computed using the *continuous SVD method* by Dieci and Elia [DE08]. The main idea is to directly integrate in terms of the SVD, where the ODE can be formulated in the logarithms of the singular values, which capture the exponential separation. Since this formulation is undefined in the case of duplicate singular values, Equation B.6, which is a system of n^2 ODEs, is first integrated using a standard ODE solver. At each time step t_i, we compute the singular value decomposition (Equation B.5), and stop integration once $\left(\sigma^1_{t_0}(t_j) - \sigma^n_{t_0}(t_j)\right)/\sigma^1_{t_0}(t_j) > 10^{-1}$ at a time step t_j.

In our experiments, we used an embedded Runge–Kutta 4/5 scheme with adaptive step size and dense output. We employed a relative tolerance of 10^{-3} and absolute tolerance of 10^{-6}. The maximum step size was chosen as the mean of the discrete timesteps $t_{i+1} - t_i$ of the candidate line, and the initial step size as a 10th of this. The dense output is used to obtain the solutions at the predefined discrete time steps t_i, which may not be reached exactly due to the adaptive step size.

For integrating over the remaining time interval, a modified ODE [DE08] is used instead. For integration of the singular values, variables $\nu_1(t), \nu_2(t)[, \nu_3(t)]$ are introduced, and for the orthogonal factors the variables $\mathbf{B}(t), \mathbf{R}(t)$. We further define (see below) matrices $\mathbf{C}(t), \mathbf{H}(t), \mathbf{K}(t)$, which are evaluated based on the state variables and simplify notation, where we write the components by subscript, i.e., A_{ij} denotes the element in the ith row and jth column of the matrix $\mathbf{A}$.

The modified ODE for the variables ν_i, with initial values at time $t = t_j$, in the 2D case are

$$\nu_1(t_j) = \sigma^2_{t_0}(t_j) - \sigma^1_{t_0}(t_j), \qquad \nu_2(t_j) = \sigma^2_{t_0}(t_j), \tag{B.7}$$

$$\frac{d}{dt}\nu_1 = C_{22}(t) - C_{11}(t), \qquad \frac{d}{dt}\nu_2 = C_{22}(t), \tag{B.8}$$

and in the 3D case

$$\nu_1(t_j) = \sigma^2_{t_0}(t_j) - \sigma^1_{t_0}(t_j), \qquad \nu_2(t_j) = \sigma^3_{t_0}(t_j) - \sigma^2_{t_0}(t_j), \qquad \nu_3(t_j) = \sigma^3_{t_0}(t_j), \tag{B.9}$$

$$\frac{d}{dt}\nu_1 = C_{22}(t) - C_{11}(t), \qquad \frac{d}{dt}\nu_2 = C_{33}(t) - C_{22}(t), \qquad \frac{d}{dt}\nu_3 = C_{33}(t). \tag{B.10}$$

The evolution of the orthogonal factors $\mathbf{B}(t), \mathbf{R}(t)$ of the SVD is given by

$$\mathbf{B}(t_j) = \mathbf{B}_{t_0}(t_j), \qquad \mathbf{R}(t_j) = \mathbf{R}_{t_0}(t_j), \tag{B.11}$$

$$\frac{d}{dt}\mathbf{B} = \mathbf{B}(t)\mathbf{H}(t), \qquad \frac{d}{dt}\mathbf{R} = \mathbf{R}(t)\mathbf{K}(t). \tag{B.12}$$

The matrices $\mathbf{C}(t), \mathbf{H}(t), \mathbf{K}(t)$ are defined as

$$\mathbf{C}(t) = \mathbf{B}(t)^\top \mathbf{J}(t)\mathbf{B}(t), \tag{B.13}$$

$$\mathbf{H}(t)^\top = -\mathbf{H}(t), \tag{B.14}$$

$$\mathbf{K}(t)^\top = -\mathbf{K}(t), \tag{B.15}$$

where the upper diagonal entries of $\mathbf{H}(t), \mathbf{K}(t)$ are for the 2D case

$$H_{12}(t) = \left(e^{\nu_1(t)}C_{12} + C_{21}\right)\left(e^{2\nu_1(t)} - 1\right)^{-1}, \tag{B.16}$$

$$K_{12}(t) = e^{\nu_1(t)}(C_{12} + C_{21})\left(e^{2\nu_1(t)} - 1\right)^{-1}, \tag{B.17}$$

and for the 3D case, with $H_{12}(t), K_{12}(t)$ as in the 2D case above,

$$H_{13}(t) = \left(e^{\nu_1(t)+\nu_2(t)}C_{12} + C_{21}\right)\left(e^{2\nu_1(t)+2\nu_2(t)} - 1\right)^{-1}, \tag{B.18}$$

$$H_{23}(t) = \left(e^{\nu_2(t)}C_{12} + C_{21}\right)\left(e^{2\nu_2(t)} - 1\right)^{-1}, \tag{B.19}$$

$$K_{13}(t) = e^{\nu_1(t)+\nu_2(t)}(C_{12} + C_{21})\left(e^{2\nu_1(t)+2\nu_2(t)} - 1\right)^{-1}, \tag{B.20}$$

$$K_{23}(t) = e^{\nu_2(t)}(C_{12} + C_{21})\left(e^{2\nu_2(t)} - 1\right)^{-1}. \tag{B.21}$$

In order to keep the matrices $\mathbf{B}(t)$ and $\mathbf{R}(t)$ orthogonal, we replace them with the Q-factor of their QR-decomposition after each numerical integration step. We enforce a consistent orientation of the column vectors by multiplying Q with the diagonal matrix $S = \operatorname{diag}(\operatorname{sign}(R_{11}), \operatorname{sign}(R_{22})[, \operatorname{sign}(R_{33})])$, i.e., we rewrite $QR = (QS)(SR)$. The singular values σ^i, i.e., the entries of $\mathbf{\Sigma}$, are obtained from the integration variables ν_i as $\sigma^i = \nu_{i+1} - \nu_i$ for $i < n$, and $\sigma^n = \nu_n$.

Bibliography

[AGL05] J. Ahrens, B. Geveci, and C. Law. “ParaView: an end-user tool for large data visualization”. *The Visualization Handbook* 717:8, 2005 (cit. on p. 17).

[Ahr+01] J. Ahrens, K. Brislawn, K. Martin, B. Geveci, C. C. Law, and M. Papka. “Large-scale data visualization using parallel data streaming”. *IEEE Computer Graphics and Applications* 21:4, 2001, pp. 34–41 (cit. on p. 17).

[Ami+19] A. Amirkhanov, I. Kosiuk, P. Szmolyan, A. Amirkhanov, G. Mistelbauer, M. E. Gröller, and R. G. Raidou. “Manylands: a journey across 4D phase space of trajectories”. *Computer Graphics Forum* 38:7, 2019, pp. 191–202 (cit. on p. 41).

[Asi85] D. Asimov. “The grand tour: a tool for viewing multidimensional data”. *SIAM Journal on Scientific and Statistical Computing* 6:1, 1985, pp. 128–143 (cit. on p. 41).

[Asi93] D. Asimov. *Notes on the topology of vector fields and flows*. Technical report. NASA Ames Research Center, RNR-93-003, 1993 (cit. on pp. 30, 71).

[BAT11] S. Barakat, N. Andrysco, and X. Tricoche. “Fast extraction of high-quality crease surfaces for visual analysis”. *Computer Graphics Forum* 30:3, 2011, pp. 961–970 (cit. on p. 20).

[BB56] R. C. Buck and E. F. Buck. *Advanced Calculus*. Tate McGraw-Hill Education, 1956 (cit. on p. 6).

[BDZG19] R. Bujack, S. Dutta, D. Zhang, and T. Günther. “Objective finite-time flow topology from flowmap expansion and contraction”. In: *Proc. TopoInVis*. Nyköping, Sweden, 2019 (cit. on pp. 40, 121).

[Ber+99] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. “The ball-pivoting algorithm for surface reconstruction”. *IEEE Transactions on Visualization and Computer Graphics* 5:4, 1999, pp. 349–359 (cit. on p. 111).

[Bey90] W.-J. Beyn. “The numerical computation of connecting orbits in dynamical systems”. *IMA Journal of Numerical Analysis* 10:3, 1990, pp. 379–405 (cit. on p. 34).

[BG20] I. Baeza Rojo and T. Günther. “Vector field topology of time-dependent flows in a steady reference frame”. *IEEE Transactions on Visualization and Computer Graphics* 26:1, 2020, pp. 280–290 (cit. on pp. 23, 35, 98, 104, 105, 117, 130–138, 140–142, 146, 147, 155).

[BG88] P. J. Barry and R. N. Goldman. “A recursive evaluation algorithm for a class of Catmull–Rom splines”. In: *Proc. 15th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. 1988, pp. 199–204 (cit. on p. 10).

[BGG18] I. Baeza Rojo, M. Gross, and T. Günther. “Visualizing the phase space of heterogeneous inertial particles in 2D flows”. *Computer Graphics Forum* 37:3, 2018, pp. 289–300 (cit. on p. 42).

[BGG20] N. Bartolovic, M. Gross, and T. Günther. “Phase space projection of dynamical systems”. *Computer Graphics Forum* 39:3, 2020, pp. 253–264 (cit. on p. 41).

[Ble+20] C. Blecha, F. Raith, A. J. Präger, T. Nagel, O. Kolditz, J. Maßmann, N. Röber, M. Böttinger, and G. Scheuermann. “Fiber surfaces for many variables”. *Computer Graphics Forum* 39:3, 2020, pp. 317–329 (cit. on p. 6).

[Bor+13] R. Borgo, J. Kehrer, D. H. Chung, E. Maguire, R. S. Laramee, H. Hauser, M. Ward, and M. Chen. “Glyph-based visualization: foundations, design guidelines, techniques and applications.” In: *Eurographics State of the Art Reports*. 2013, pp. 39–63 (cit. on p. 9).

[BP02] D. Bauer and R. Peikert. “Vortex tracking in scale-space”. In: *Proc. Symposium on Data Visualisation*. 2002, 233–ff (cit. on p. 111).

[BPB14] H. Bhatia, V. Pascucci, and P.-T. Bremer. “The natural Helmholtz–Hodge decomposition for open-boundary flow analysis”. *IEEE Transactions on Visualization and Computer Graphics* 20:11, 2014, pp. 1566–1578 (cit. on p. 35).

[BPKB14] H. Bhatia, V. Pascucci, R. M. Kirby, and P.-T. Bremer. “Extracting features from time-dependent vector fields using internal reference frames”. *Computer Graphics Forum* 33:3, 2014, pp. 21–30 (cit. on p. 35).

[Bra+12] A. Brambilla, R. Carnecky, R. Peikert, I. Viola, and H. Hauser. “Illustrative flow visualization: state of the art, trends and challenges”. In: *Eurographics State of the Art Reports*. 2012 (cit. on p. 9).

[Bra86] R. N. Bracewell. *The Fourier Transform and its Applications*. 2. ed., rev. McGraw-Hill Series in Electrical Engineering: Circuits and Systems. 1986 (cit. on p. 12).

[BSDW12] S. Bachthaler, F. Sadlo, C. Dachsbacher, and D. Weiskopf. “Space-time visualization of dynamics in Lagrangian coherent structures of time-dependent 2D vector fields”. In: *Proc. International Conference on Information Visualization Theory and Applications (IVAPP)*. 2012, pp. 573–583 (cit. on p. 129).

[Buj+19] R. Bujack, S. Dutta, I. Baeza Rojo, D. Zhang, and T. Günther. “Objective finite-time saddles and their connection to FTLE”. In: *Short Paper Proc. EuroVis*. 2019, pp. 49–53 (cit. on p. 40).

[Buj+20] R. Bujack, L. Yan, I. Hotz, C. Garth, and B. Wang. “State of the art in time-dependent flow topology: interpreting physical meaningfulness through mathematical properties”. *Computer Graphics Forum* 39:3, 2020, pp. 811–835 (cit. on pp. 9, 114).

[BW09] M. Branicki and S. Wiggins. “An adaptive method for computing invariant manifolds in non-autonomous, three-dimensional dynamical systems”. *Physica D: Nonlinear Phenomena* 238:16, 2009, pp. 1625–1657 (cit. on pp. 40, 98, 99, 102, 119, 138, 146).

[BW10] M. Branicki and S. Wiggins. “Finite-time Lagrangian transport analysis: stable and unstable manifolds of hyperbolic trajectories and finite-time Lyapunov exponents”. *Nonlinear Processes in Geophysics* 17:1, 2010, pp. 1–36 (cit. on pp. 40, 100, 102, 108, 114, 143).

[BWC04] P. Bhaniramka, R. Wenger, and R. Crawfis. “Isosurface construction in any dimension using convex hulls”. *IEEE Transactions on Visualization and Computer Graphics* 10:2, 2004, pp. 130–141 (cit. on pp. 41, 50).

[Cal10] J. J. Callahan. *Advanced Calculus: A Geometric View*. Springer Science & Business Media, 2010 (cit. on p. 6).

[Car+15] H. Carr, Z. Geng, J. Tierny, A. Chattopadhyay, and A. Knoll. “Fiber surfaces: generalizing isosurfaces to bivariate data”. In: *Computer Graphics Forum*. Vol. 34. 3. 2015, pp. 241–250 (cit. on p. 6).

[CFHH09] A. Chu, C.-W. Fu, A. Hanson, and P.-A. Heng. “GL4D: a GPU-based architecture for interactive 4D visualization”. *IEEE Transactions on Visualization and Computer Graphics* 15:6, 2009, pp. 1587–1594 (cit. on p. 41).

[Chi+10] H. Childs, D. Pugmire, S. Ahern, B. Whitlock, M. Howison, G. H. Weber, E. W. Bethel, et al. “Extreme scaling of production visualization software on diverse architectures”. *IEEE Computer Graphics and Applications* 30:3, 2010, pp. 22–31 (cit. on p. 17).

[CK14] F. Colonius and W. Kliemann. *Dynamical Systems and Linear Algebra*. Vol. 158. American Mathematical Society, 2014 (cit. on pp. 8, 26).

[CL93] B. Cabral and L. C. Leedom. “Imaging vector fields using line integral convolution”. In: *Proc. 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. 1993, pp. 263–270 (cit. on p. 9).

[Cop78] W. A. Coppel. *Dichotomies in Stability Theory*. Vol. 629. Springer, 1978 (cit. on pp. 40, 100).

[CR74] E. Catmull and R. Rom. “A class of local interpolating splines”. In: *Computer aided geometric design*. Elsevier, 1974, pp. 317–326 (cit. on p. 10).

[Dam98] J. Damon. “Generic structure of two-dimensional images under Gaussian blurring”. *SIAM Journal on Applied Mathematics* 59:1, 1998, pp. 97–138 (cit. on p. 20).

[DE08] L. Dieci and C. Elia. “SVD algorithms to approximate spectra of dynamical systems”. *Mathematics and Computers in Simulation* 79:4, 2008, pp. 1235–1254 (cit. on pp. 105, 156).

[Deu11] P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Vol. 35. Springer Science & Business Media, 2011 (cit. on p. 31).

[DFIM15] L. De Floriani, U. Fugacci, F. Iuricich, and P. Magillo. “Morse complexes for shape segmentation and homological analysis: discrete models and algorithms”. In: *Computer Graphics Forum*. Vol. 34. 2. 2015, pp. 761–785 (cit. on p. 7).

[DL14] C. Dong and Y. Lan. “A variational approach to connecting orbits in nonlinear dynamical systems”. *Physics Letters A* 378:9, 2014, pp. 705–712 (cit. on p. 34).

[DP80] J. Dormand and P. Prince. “A family of embedded Runge–Kutta formulae”. *Journal of Computational and Applied Mathematics* 6:1, 1980, pp. 19–26 (cit. on p. 12).

[DSL90] D. Degani, A. Seginer, and Y. Levy. “Graphical visualization of vortical flows by means of helicity”. *AIAA Journal* 28:8, 1990, pp. 1347–1352 (cit. on p. 18).

[DV99] W. De Leeuw and R. Van Liere. “Collapsing flow topology using area metrics”. In: *Proc. IEEE Visualization*. 1999, pp. 349–542 (cit. on p. 34).

[Ebe+94] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. “Ridges for image analysis”. *Journal of Mathematical Imaging and Vision* 4:4, 1994, pp. 353–373 (cit. on pp. 7, 18, 19, 48).

[ELZ00] H. Edelsbrunner, D. Letscher, and A. Zomorodian. “Topological persistence and simplification”. In: *Proc. 41st Annual Symposium on Foundations of Computer Science*. 2000, pp. 454–463 (cit. on p. 7).

[Eng+04] K. Engel, M. Hadwiger, J. M. Kniss, A. E. Lefohn, C. R. Salama, and D. Weiskopf. “Real-time volume graphics”. In: *ACM SIGGRAPH Course Notes*. 2004, 29–es (cit. on pp. 6, 17).

[ESRT13] J. M. Esturo, M. Schulze, C. Röss, and H. Theisel. “Poisson-based tools for flow visualization”. In: *Proc. IEEE Pacific Visualization Symposium*. 2013, pp. 241–248 (cit. on p. 15).

[FD91] M. J. Friedman and E. J. Doedel. “Numerical computation and continuation of invariant manifolds connecting fixed points”. *SIAM Journal on Numerical Analysis* 28:3, 1991, pp. 789–808 (cit. on p. 34).

[FGRT17] A. Friederici, T. Günther, C. Rössl, and H. Theisel. “Finite time steady vector field topology – theoretical foundation and 3D case”. In: *Proc. Vision, Modeling and Visualization*. 2017 (cit. on p. 113).

[FP01] J. D. Furst and S. M. Pizer. “Marching ridges”. In: *Proc. IASTED International Conference on Signal and Image Processing*. 2001, pp. 22–26 (cit. on pp. 19, 52).

[Fuc+08] R. Fuchs, R. Peikert, H. Hauser, F. Sadlo, and P. Muigg. “Parallel vectors criteria for unsteady flow vortices”. *IEEE Transactions on Visualization and Computer Graphics* 14:3, 2008, pp. 615–626 (cit. on p. 18).

[Gar+04] C. Garth, X. Tricoche, T. Salzbrunn, T. Bobach, and G. Scheuermann. "Surface techniques for vortex visualization." In: *Proc. Eurographics / IEEE VGTC Symposium on Visualization*. Vol. 4. 2004, pp. 155–164 (cit. on p. 15).

[Gar+08] C. Garth, H. Krishnan, X. Tricoche, T. Bobach, and K. I. Joy. "Generation of accurate integral surfaces in time-dependent vector fields". *IEEE Transactions on Visualization and Computer Graphics* 14:6, 2008, pp. 1404–1411 (cit. on p. 15).

[Gar+09] C. Garth, G.-S. Li, X. Tricoche, C. D. Hansen, and H. Hagen. "Visualization of coherent structures in transient 2D flows". In: *Topology-Based Methods in Visualization II*. Springer, 2009, pp. 1–13 (cit. on p. 36).

[GG17] T. Günther and M. Gross. "Flow-induced inertial steady vector field topology". *Computer Graphics Forum* 36:2, 2017, pp. 143–152 (cit. on pp. 42, 71, 94).

[GGT17] T. Günther, M. Gross, and H. Theisel. "Generic objective vortices for flow visualization". *ACM Transactions on Graphics* 36:4, 2017, 141:1–141:11 (cit. on pp. 22, 23, 129).

[GGTH07] C. Garth, F. Gerhardt, X. Tricoche, and H. Hagen. "Efficient computation and visualization of coherent structures in fluid flow applications". *IEEE Transactions on Visualization and Computer Graphics* 13:6, 2007, pp. 1464–71 (cit. on p. 36).

[GHWG14] S. Grottel, J. Heinrich, D. Weiskopf, and S. Gumhold. "Visual analysis of trajectories in multi-dimensional state spaces". *Computer Graphics Forum* 33:6, 2014, pp. 310–321 (cit. on p. 41).

[GJ10] C. Garth and K. I. Joy. "Fast, memory-efficient cell location in unstructured grids for visualization". *IEEE Transactions on Visualization and Computer Graphics* 16:6, 2010, pp. 1541–1550 (cit. on p. 10).

[GNS05] H. Gunawan, O. Neswan, and W. Setya-Budhi. "A formula for angles between subspaces of inner product spaces". *Contributions to Algebra and Geometry* 46:2, 2005, pp. 311–320 (cit. on p. 56).

[GRT18] T. Gerrits, C. Rössl, and H. Theisel. "An approximate parallel vectors operator for multiple vector fields". *Computer Graphics Forum* 37:3, 2018, pp. 315–326 (cit. on p. 18).

[GST16] T. Günther, M. Schulze, and H. Theisel. “Rotation invariant vortices for flow visualization”. *IEEE Transactions on Visualization and Computer Graphics* 22:1, 2016, pp. 817–826 (cit. on pp. 21, 121).

[GT15] T. Günther and H. Theisel. “Finite-time mass separation for comparative visualizations of inertial particles”. *Computer Graphics Forum* 34:3, 2015, pp. 471–480 (cit. on p. 42).

[GT16a] T. Günther and H. Theisel. “Backward finite-time Lyapunov exponents in inertial flows”. *IEEE Transactions on Visualization and Computer Graphics* 23:1, 2016, pp. 970–979 (cit. on p. 42).

[GT16b] T. Günther and H. Theisel. “Inertial steady 2D vector field topology”. *Computer Graphics Forum* 35:2, 2016, pp. 455–466 (cit. on p. 42).

[GT18a] T. Günther and H. Theisel. “Objective vortex corelines of finite-sized objects in fluid flows”. *IEEE Transactions on Visualization and Computer Graphics*, 2018 (cit. on pp. 18, 67).

[GT18b] T. Günther and H. Theisel. “The state of the art in vortex extraction”. *Computer Graphics Forum* 37:6, 2018, pp. 149–173 (cit. on p. 24).

[GT20] T. Günther and H. Theisel. “Hyper-objective vortices”. *IEEE Transactions on Visualization and Computer Graphics* 26:3, 2020, pp. 1532–1547 (cit. on pp. 23, 105, 146, 147, 155).

[GTS04] C. Garth, X. Tricoche, and G. Scheuermann. “Tracking of vector field singularities in unstructured 3D time-dependent datasets”. In: *Proc. IEEE Conference on Visualization*. 2004, pp. 329–336 (cit. on pp. 35, 71, 94).

[Gün16] T. Günther. “Opacity optimization and inertial particles in flow visualization”. PhD thesis. 2016 (cit. on p. 42).

[Had+17] A. Hadjighasem, M. Farazmand, D. Blazevski, G. Froyland, and G. Haller. “A critical comparison of Lagrangian methods for coherent structure detection”. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27:5, 2017, p. 053104 (cit. on p. 36).

[Hal00] G. Haller. “Finding finite-time invariant manifolds in two-dimensional velocity fields”. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 10:1, 2000, pp. 99–108 (cit. on pp. 38, 97, 100, 102).

[Hal01] G. Haller. “Distinguished material surfaces and coherent structures in three-dimensional fluid flows”. *Physica D: Nonlinear Phenomena* 149:4, 2001, pp. 248–277 (cit. on pp. 36, 102, 115).

[Hal15] G. Haller. “Lagrangian coherent structures”. *Annual Review of Fluid Mechanics* 47, 2015, pp. 137–162 (cit. on p. 102).

[Hal21] G. Haller. “Can vortex criteria be objectivized?” *Journal of Fluid Mechanics* 908, 2021 (cit. on p. 23).

[HC93] A. J. Hanson and R. A. Cross. “Interactive visualization methods for four dimensions”. In: *Proc. IEEE Conference on Visualization*. 1993, pp. 196–203 (cit. on p. 40).

[Hei+16] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. “A survey of topology-based methods in visualization”. *Computer Graphics Forum* 35:3, 2016, pp. 643–667 (cit. on p. 6).

[HH89] J. Helman and L. Hesselink. “Representation and display of vector field topology in fluid flow data sets”. *IEEE Computer* 22:8, 1989, pp. 27–36 (cit. on pp. 26, 97).

[HH91] J. L. Helman and L. Hesselink. “Visualizing vector field topology in fluid flows”. *IEEE Computer Graphics and Applications* 11:3, 1991, pp. 36–46 (cit. on p. 26).

[HH92] A. J. Hanson and P. A. Heng. “Illuminating the fourth dimension”. *IEEE Computer Graphics and Applications* 12:4, 1992, pp. 54–62 (cit. on p. 40).

[HHS93] H.-C. Hege, T. Höllerer, and D. Stalling. “Volume rendering-mathematicals models and algorithmic aspects”, 1993 (cit. on p. 6).

[HIM99] A. J. Hanson, K. I. Ishkov, and J. H. Ma. *Meshview: Visualizing the fourth dimension*. Technical report. Indiana University, 1999 (cit. on p. 41).

[HJ11] C. D. Hansen and C. R. Johnson. *Visualization Handbook*. Elsevier, 2011 (cit. on p. 6).

[HMTR18] M. Hadwiger, M. Mlejnek, T. Theußl, and P. Rautek. “Time-dependent flow seen through approximate observer Killing fields”. *IEEE Transactions on Visualization and Computer Graphics* 25:1, 2018, pp. 1257–1266 (cit. on p. 23).

[HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I, Nonstiff Problems*. Springer, 1993 (cit. on pp. 12, 13).

[Hol91] S. R. Hollasch. “Four-space visualization of 4D objects”. MA thesis. Arizona State University, 1991 (cit. on pp. 40, 75).

[HRS18] L. Hofmann, B. Rieck, and F. Sadlo. “Visualization of 4D vector field topology”. *Computer Graphics Forum* 37:3, 2018, pp. 301–313 (cit. on pp. 2, 3).

[HS11] G. Haller and T. Sapsis. “Lagrangian coherent structures and the smallest finite-time Lyapunov exponent”. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 21:2, 2011, p. 023115 (cit. on p. 38).

[HS19] L. Hofmann and F. Sadlo. “The dependent vectors operator”. *Computer Graphics Forum* 38:3, 2019, pp. 261–272 (cit. on pp. 2, 3).

[HS20] L. Hofmann and F. Sadlo. “Extraction of distinguished hyperbolic trajectories for 2D time-dependent vector field topology”. *Computer Graphics Forum* 39:3, 2020, pp. 303–315 (cit. on pp. 2, 3).

[HS21] L. Hofmann and F. Sadlo. “Local extraction of 3D time-dependent vector field topology”. *Computer Graphics Forum* 40:3, 2021, pp. 111–122 (cit. on pp. 2, 3).

[Hul90] J. P. Hultquist. “Interactive numerical flow visualization using stream surfaces”. *Computing Systems in Engineering* 1:2-4, 1990, pp. 349–353 (cit. on p. 14).

[Hul92] J. P. Hultquist. “Constructing stream surfaces in steady 3D vector fields”. In: *Proc. IEEE Conference on Visualization*. 1992, pp. 171–178 (cit. on pp. 14, 80, 107, 109).

[Ins85] A. Inselberg. “The plane with parallel coordinates”. *The Visual Computer* 1:2, 1985, pp. 69–91 (cit. on p. 41).

[ISW02] K. Ide, D. Small, and S. Wiggins. “Distinguished hyperbolic trajectories in time-dependent fluid flows: analytical and computational approach for velocity fields defined as data sets”. *Nonlinear Processes in Geophysics* 9, 2002, pp. 237–263 (cit. on pp. 40, 98–102, 104, 121, 122, 143, 146).

[JCWD14] T. Ju, M. Cheng, X. Wang, and Y. Duan. “A robust parity test for extracting parallel vectors in 3D”. *IEEE Transactions on Visualization and Computer Graphics* 20:12, 2014, pp. 2526–2534 (cit. on p. 18).

[JH18] J. Jankowai and I. Hotz. “Feature level-sets: generalizing iso-surfaces to multi-variate data”. *IEEE Transactions on Visualization and Computer Graphics* 26:2, 2018, pp. 1308–1319 (cit. on p. 6).

[JSW03] N. Ju, D. Small, and S. Wiggins. “Existence and computation of hyperbolic trajectories of aperiodically time dependent vector fields and their approximations”. *International Journal of Bifurcation and Chaos* 13:06, 2003, pp. 1449–1457 (cit. on pp. 40, 99, 102, 125).

[Jun+17] P. Jung, P. Hausner, L. Pilz, J. Stern, C. Euler, M. Riemer, and F. Sadlo. “Tumble-vortex core line extraction”. In: *Proc. SIBGRAPI WVIS*. 2017 (cit. on p. 18).

[Kan00] E. Kandogan. “Star coordinates: a multi-dimensional visualization technique with uniform treatment of dimensions”. In: *Proc. IEEE Information Visualization Symposium*. Vol. 650. 2000, p. 22 (cit. on p. 41).

[Kas+09] J. Kasten, C. Petz, I. Hotz, B. R. Noack, and H.-C. Hege. “Localized finite-time Lyapunov exponent for unsteady flow analysis.” In: *Proc. Vision, Modeling and Visualization*. 2009, pp. 265–276 (cit. on pp. 37, 98, 109).

[Ken98] D. N. Kenwright. “Automatic detection of open and closed separation and attachment lines”. In: *Proc. IEEE Visualization*. 1998, pp. 151–158 (cit. on pp. 34, 146).

[KGJ09] H. Krishnan, C. Garth, and K. Joy. “Time and streak surfaces for flow visualization in large time-varying data sets”. *IEEE Transactions on Visualization and Computer Graphics* 15:6, 2009, pp. 1267–1274 (cit. on pp. 16, 108).

[KHL99] D. N. Kenwright, C. Henze, and C. Levit. “Feature extraction of separation and attachment lines”. *IEEE Transactions on Visualization and Computer Graphics* 5:2, 1999, pp. 135–144 (cit. on pp. 18, 34).

[Kin+18] G. Kindlmann, C. Chiw, T. Huynh, A. Gyulassy, J. Reppy, and P.-T. Bremer. “Rendering and extracting extremal features in 3D fields”. *Computer Graphics Forum* 37:3, 2018, pp. 525–536 (cit. on pp. 10, 17, 20).

[KKH02] J. Kniss, G. Kindlmann, and C. Hansen. “Multidimensional transfer functions for interactive volume rendering”. *IEEE Transactions on Visualization and Computer Graphics* 8:3, 2002, pp. 270–285 (cit. on p. 6).

[Knu11] D. E. Knuth. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*. Addison-Wesley Professional, 2011 (cit. on p. 153).

[KRRS14] J. Kasten, J. Reininghaus, W. Reich, and G. Scheuermann. "Toward the extraction of saddle periodic orbits". In: *Topological Methods in Data Analysis and Visualization III*. Springer, 2014, pp. 55–69 (cit. on pp. 31, 94).

[KTCG16] P. Klacansky, J. Tierny, H. Carr, and Z. Geng. "Fast and exact fiber surfaces for tetrahedral meshes". *IEEE Transactions on Visualization and Computer Graphics* 23:7, 2016, pp. 1782–1795 (cit. on p. 6).

[KTW06] G. Kindlmann, X. Tricoche, and C.-F. Westin. "Anisotropy creases delineate white matter structure in diffusion tensor MRI". In: *Proc. International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2006, pp. 126–133 (cit. on p. 20).

[KTW07] G. Kindlmann, X. Tricoche, and C.-F. Westin. "Delineating white matter structure in diffusion tensor MRI with anisotropy creases". *Medical Image Analysis* 11:5, 2007, pp. 492–502 (cit. on p. 10).

[Kuh+14] A. Kuhn, W. Engelke, C. Rössl, M. Hadwiger, and H. Theisel. "Time line cell tracking for the approximation of Lagrangian coherent structures with subgrid accuracy". *Computer Graphics Forum* 33:1, 2014, pp. 222–234 (cit. on p. 36).

[Lar+04] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. "The state of the art in flow visualization: dense and texture-based techniques". *Computer Graphics Forum* 23:2, 2004, pp. 203–221 (cit. on p. 9).

[LC04] Y. Lan and P. Cvitanović. "Variational method for finding periodic orbits in a general flow". *Physical Review E* 69:1, 2004, p. 016217 (cit. on p. 32).

[LC87] W. E. Lorensen and H. E. Cline. "Marching cubes: a high resolution 3D surface construction algorithm". In: *Proc. 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. Vol. 21. 4. 1987, pp. 163–169 (cit. on pp. 6, 19, 41, 50, 74).

[LHZP07] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post. "Topology-based flow visualization: The state of the art". In: *Topology-Based Methods in Visualization*. Springer, 2007, pp. 1–19 (cit. on p. 9).

[LKG98] H. Löffelmann, T. Kučera, and E. Gröller. "Visualizing Poincaré maps together with the underlying flow". In: *Mathematical Visualization*. Springer, 1998, pp. 315–328 (cit. on p. 30).

[LLSV99] A. Lopez, F. Lumbreras, J. Serrat, and J. Villanueva. “Evaluation of methods for ridge and valley detection”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21:4, 1999, pp. 327–335 (cit. on p. 7).

[Lou01] P. Lounesto. *Clifford Algebras and Spinors*. Vol. 286. Cambridge University Press, 2001 (cit. on p. 47).

[Lug79] H. J. Lugt. “The dilemma of defining a vortex”. In: *Recent Developments in Theoretical and Experimental Fluid Mechanics*. Springer, 1979, pp. 309–321 (cit. on p. 22).

[Max70] J. C. Maxwell. “On hills and dales”. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 40:269, 1870, pp. 421–427 (cit. on p. 7).

[MBC93] N. Max, B. Becker, and R. Crawfis. “Flow volumes for interactive vector field visualization”. In: *Proc. IEEE Conference on Visualization*. 1993, pp. 19–24 (cit. on pp. 14, 80).

[MBES16] G. M. Machado, S. Boblest, T. Ertl, and F. Sadlo. “Space-time bifurcation lines for extraction of 2D Lagrangian coherent structures”. *Computer Graphics Forum* 35:3, 2016, pp. 91–100 (cit. on pp. 26, 34, 39, 94, 98, 104, 106, 114, 128, 130, 134, 146).

[McL+10] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. “Over two decades of integration-based, geometric flow visualization”. *Computer Graphics Forum* 29:6, 2010, pp. 1807–1829 (cit. on p. 9).

[Mei86] L. Meirovitch. *Elements of Vibration Analysis*. 2nd ed. McGraw-Hill Science/Engineering/Math, 1986 (cit. on p. 60).

[MLZ09] T. McLoughlin, R.S. Laramee, and E. Zhang. “Easy integral surfaces: a fast, quad-based stream and path surface algorithm”. In: *Proc. Computer Graphics International Conference*. 2009, pp. 73–82 (cit. on p. 14).

[Mor95] B. S. Morse. *Computation of Object Cores from Grey-level Images*. The University of North Carolina at Chapel Hill, 1995 (cit. on p. 19).

[MSE13] G. M. Machado, F. Sadlo, and T. Ertl. “Local extraction of bifurcation lines”. In: *Proc. Vision, Modeling and Visualization*. 2013, pp. 17–24 (cit. on pp. 18, 24–26, 32, 34, 94, 102, 111, 114, 121–123, 129–132, 135, 143, 146).

[MSE14] G. M. Machado, F. Sadlo, and T. Ertl. “Image-based streamsurfaces”. In: *Proc. SIBGRAPI Conference on Graphics, Patterns and Images*. IEEE. 2014, pp. 343–350 (cit. on p. 15).

[MSW04] A. Mancho, D. Small, and S. Wiggins. “Computation of hyperbolic trajectories and their stable and unstable manifolds for oceanographic flows represented as data sets”. *Nonlinear Processes in Geophysics* 11:1, 2004, pp. 17–33 (cit. on p. 40).

[MW98] N. Malhotra and S. Wiggins. “Geometric structures, lobe dynamics, and Lagrangian transport in flows with aperiodic time-dependence, with applications to Rossby wave flow”. *Journal of Nonlinear Science* 8:4, 1998, pp. 401–456 (cit. on pp. 117, 147).

[MWCM13] A. M. Mancho, S. Wiggins, J. Curbelo, and C. Mendoza. “Lagrangian descriptors: a method for revealing phase space structures of general time dependent dynamical systems”. *Communications in Nonlinear Science and Numerical Simulation* 18:12, 2013, pp. 3530–3557 (cit. on p. 40).

[NA18] L. G. Nonato and M. Aupetit. “Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment”. *IEEE Transactions on Visualization and Computer Graphics* 25:8, 2018, pp. 2650–2673 (cit. on p. 41).

[Nol67] A. M. Noll. “A computer technique for displaying n-dimensional hyperobjects”. *Communications of the ACM* 10:8, 1967, pp. 469–473 (cit. on pp. 40, 75).

[OHH15] K. Onu, F. Huhn, and G. Haller. “LCS tool: a computational platform for Lagrangian coherent structures”. *Journal of Computational Science* 7, 2015, pp. 26–36 (cit. on p. 36).

[ORT18a] T. Oster, C. Rössl, and H. Theisel. “Core lines in 3D second-order tensor fields”. *Computer Graphics Forum* 37:3, 2018, pp. 327–337 (cit. on pp. 18, 67).

[ORT18b] T. Oster, C. Rössl, and H. Theisel. “The parallel eigenvectors operator”. In: *Proc. Vision, Modeling and Visualization*. 2018, pp. 39–46 (cit. on pp. 18, 67, 145).

[Pag+11] C. Pagot, D. Osmari, F. Sadlo, D. Weiskopf, T. Ertl, and J. Comba. “Efficient parallel vectors feature extraction from higher-order data”. *Computer Graphics Forum* 30:3, 2011, pp. 751–760 (cit. on p. 18).

[PBK10] H. Piringer, W. Berger, and J. Krasser. “Hypermoval: Interactive visual validation of regression models for real-time simulation”. *Computer Graphics Forum* 29:3, 2010, pp. 983–992 (cit. on p. 41).

[PC87] A. E. Perry and M. S. Chong. “A description of eddying motions and flow patterns using critical-point concepts”. *Annual Review of Fluid Mechanics* 19:1, 1987, pp. 125–155 (cit. on pp. 25, 49).

[Pob+11] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matković, and H. Hauser. “The state of the art in topology-based visualization of unsteady flow”. *Computer Graphics Forum* 30:6, 2011, pp. 1789–1811 (cit. on p. 9).

[Pop04] S. Popinet. “Free computational fluid dynamics”. *ClusterWorld* 2:6, 2004. URL: http://gfs.sf.net/ (cit. on pp. 128, 140).

[Pos+03] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. “The state of the art in flow visualisation: feature extraction and tracking”. *Computer Graphics Forum* 22:4, 2003, pp. 775–792 (cit. on p. 9).

[PR99] R. Peikert and M. Roth. “The parallel vectors operator: a vector field visualization primitive”. In: *Proc. IEEE Visualization*. 1999, pp. 263–270 (cit. on pp. 17, 18, 50, 55, 56, 145).

[PS08] R. Peikert and F. Sadlo. “Height ridge computation and filtering for visualization”. In: *Proc. IEEE Pacific Visualization Symposium*. 2008, pp. 119–126 (cit. on pp. 7, 18–20).

[PS09a] R. Peikert and F. Sadlo. “Flow topology beyond skeletons: visualization of features in recirculating flow”. In: *Topology-Based Methods in Visualization II*. Springer, 2009, pp. 145–160 (cit. on p. 34).

[PS09b] R. Peikert and F. Sadlo. “Topologically relevant stream surfaces for flow visualization”. In: *Proc. 25th Spring Conference on Computer Graphics*. 2009, pp. 35–42 (cit. on p. 15).

[Rau+20] P. Rautek, M. Mlejnek, J. Beyer, J. Troidl, H. Pfister, T. Theußl, and M. Hadwiger. “Objective observer-relative flow visualization in curved spaces for unsteady 2D geophysical flows”. *IEEE Transactions on Visualization and Computer Graphics* 27:2, 2020, pp. 283–293 (cit. on p. 23).

[RHTE99] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. “Interactive exploration of volume line integral convolution based on 3D-texture mapping”. In: *Proc. IEEE Visualization*. 1999, pp. 233–528 (cit. on p. 9).

[RKWH11] J. Reininghaus, J. Kasten, T. Weinkauf, and I. Hotz. "Efficient computation of combinatorial feature flow fields". *IEEE Transactions on Visualization and Computer Graphics* 18:9, 2011, pp. 1563–1573 (cit. on p. 21).

[Rot00] M. Roth. "Automatic extraction of vortex core lines and other line-type features for scientific visualization". PhD thesis. ETH Zurich, No. 13673, 2000 (cit. on pp. 17, 18, 25, 49).

[RP98] M. Roth and R. Peikert. "A higher-order method for finding vortex core lines". In: *Proc. IEEE Visualization*. 1998, pp. 143–150 (cit. on pp. 18, 145).

[Sag+17] A. Sagristà, S. Jordan, A. Just, F. Dias, L. G. Nonato, and F. Sadlo. "Topological analysis of inertial dynamics". *IEEE Transactions on Visualization and Computer Graphics* 23:1, 2017, pp. 950–959 (cit. on pp. 42, 94, 146).

[SBHH15] C. R. Short, D. Blazevski, K. C. Howell, and G. Haller. "Stretching in phase space and applications in general nonautonomous multi-body problems". *Celestial Mechanics and Dynamical Astronomy* 122:3, 2015, pp. 213–238 (cit. on p. 42).

[Sch+97] G. Scheuermann, H. Hagen, H. Kruger, M. Menzel, and A. Rockwood. "Visualization of higher order singularities in vector fields". In: *Proc. IEEE Visualization*. 1997, pp. 67–74 (cit. on p. 34).

[SGRT12] M. Schulze, T. Germer, C. Rössl, and H. Theisel. "Stream surface parametrization by flow-orthogonal front lines". *Computer Graphics Forum* 31:5, 2012, pp. 1725–1734 (cit. on p. 15).

[SH95] D. Sujudi and R. Haimes. "Identification of swirling flow in 3-D vector fields". In: *Proc. 12th AIAA Computational Fluid Dynamics Conference*. 1995 (cit. on pp. 18, 24, 48, 110).

[Sha85] L. F. Shampine. "Interpolation for Runge–Kutta methods". *SIAM Journal on Numerical Analysis* 22:5, 1985, pp. 1014–1027 (cit. on p. 13).

[SHJK00] G. Scheuermann, B. Hamann, K. I. Joy, and W. Kollmann. "Visualizing local vector field topology". *Journal of Electronic Imaging* 9:4, 2000, pp. 356–367 (cit. on p. 34).

[SHTG14] W. Schlei, K. C. Howell, X. Tricoche, and C. Garth. "Enhanced visualization and autonomous extraction of Poincaré map topology". *The Journal of the Astronautical Sciences* 61:2, 2014, pp. 170–197 (cit. on p. 32).

[Sim90] C. Simó. “On the analytical and numerical approximation of invariant manifolds”. *Les Méthodes Modernes de la Mécanique Céleste. Modern methods in celestial mechanics*, 1990, pp. 285–329 (cit. on p. 31).

[SJS20] A. Sagristà, S. Jordan, and F. Sadlo. “Visual analysis of the finite-time Lyapunov exponent”. *Computer Graphics Forum* 39:3, 2020, pp. 331–342 (cit. on p. 147).

[SK97] H.-W. Shen and D. L. Kao. “UFLIC: a line integral convolution algorithm for visualizing unsteady flows”. In: *Proc. IEEE Visualization*. 1997, pp. 317–322 (cit. on p. 9).

[SKMR98] G. Scheuermann, H. Kruger, M. Menzel, and A. P. Rockwood. “Visualizing nonlinear vector field topology”. *IEEE Transactions on Visualization and Computer Graphics* 4:2, 1998, pp. 109–116 (cit. on p. 34).

[SLM05] S. C. Shadden, F. Lekien, and J. E. Marsden. “Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows”. *Physica D: Nonlinear Phenomena* 212:3-4, 2005, pp. 271–304 (cit. on pp. 36, 63, 97, 116, 118).

[SML98] W. Schroeder, K. M. Martin, and W. E. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice-Hall, 1998 (cit. on p. 10).

[SP07] F. Sadlo and R. Peikert. “Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction”. *IEEE Transactions on Visualization and Computer Graphics* 13:6, 2007, pp. 1456–1463 (cit. on p. 36).

[SP09] F. Sadlo and R. Peikert. “Visualizing Lagrangian coherent structures and comparison to vector field topology”. In: *Topology-Based Methods in Visualization II*. Springer, 2009, pp. 15–29 (cit. on p. 36).

[SRGT12] M. Schulze, C. Rössl, T. Germer, and H. Theisel. “As-perpendicular-as-possible surfaces for flow visualization”. In: *Proc. IEEE Pacific Visualization Symposium*. 2012, pp. 153–160 (cit. on p. 15).

[SRP11] F. Sadlo, A. Rigazzi, and R. Peikert. “Time-dependent visualization of Lagrangian coherent structures by grid advection”. In: *Topological Methods in Data Analysis and Visualization*. Springer, 2011, pp. 151–165 (cit. on p. 36).

[SRWS10] D. Schneider, W. Reich, A. Wiebel, and G. Scheuermann. "Topology aware stream surfaces". *Computer Graphics Forum* 29:3, 2010, pp. 1153–1161 (cit. on p. 15).

[STS09] T. Schultz, H. Theisel, and H.-P. Seidel. "Crease surfaces: from theory to extraction and application to diffusion tensor MRI". *IEEE Transactions on Visualization and Computer Graphics* 16:1, 2009, pp. 109–119 (cit. on pp. 11, 20, 119, 145).

[SW10] F. Sadlo and D. Weiskopf. "Time-dependent 2-D vector field topology: an approach inspired by Lagrangian coherent structures". *Computer Graphics Forum* 29:1, 2010, pp. 88–100 (cit. on pp. 38, 97, 119, 120).

[SWS09] D. Schneider, A. Wiebel, and G. Scheuermann. "Smooth stream surfaces of fourth order precision". *Computer Graphics Forum* 28:3, 2009, pp. 871–878 (cit. on p. 15).

[SWTH07] J. Sahner, T. Weinkauf, N. Teuber, and H.-C. Hege. "Vortex and strain skeletons in Eulerian and Lagrangian frames". *IEEE Transactions on Visualization and Computer Graphics* 13:5, 2007, pp. 980–990 (cit. on p. 7).

[TGS06] X. Tricoche, C. Garth, and G. Scheuermann. "Fast and robust extraction of separation line features". In: *Scientific Visualization: The Visual Extraction of Knowledge from Data*. Springer, 2006, pp. 249–263 (cit. on p. 112).

[TGS11] X. Tricoche, C. Garth, and A. Sanderson. "Visualization of topological structures in area-preserving maps". *IEEE Transactions on Visualization and Computer Graphics* 17:12, 2011, pp. 1765–1774 (cit. on p. 30).

[The+05] H. Theisel, J. Sahner, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. "Extraction of parallel vector surfaces in 3D time-dependent fields and application to vortex core line tracking". In: *Proc. IEEE Visualization*. 2005, pp. 631–638 (cit. on pp. 21, 111, 145).

[The+21] H. Theisel, M. Hadwiger, P. Rautek, T. Theußl, and T. Günther. "Vortex criteria can be objectivized by unsteadiness minimization". *2106.16169*, 2021. arXiv: `2106.16169 [physics.flu-dyn]` (cit. on p. 24).

[Tie+17] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. "The topology toolkit". *IEEE Transactions on Visualization and Computer Graphics* 24:1, 2017, pp. 832–842 (cit. on p. 130).

[TN04] C. Truesdell and W. Noll. *The Non-Linear Field Theories of Mechanics*. Springer, 2004 (cit. on p. 22).

[TS03] H. Theisel and H.-P. Seidel. “Feature flow fields”. In: *Proc. Symposium on Data Visualisation*. 2003, pp. 141–148 (cit. on pp. 20, 21, 35, 67, 145).

[TSH00] X. Tricoche, G. Scheuermann, and H. Hagen. “A topology simplification method for 2D vector fields”. In: *Proc. IEEE Visualization*. 2000, pp. 359–366 (cit. on p. 34).

[TSH01a] X. Tricoche, G. Scheuermann, and H. Hagen. “Continuous topology simplification of planar vector fields”. In: *Proc. IEEE Visualization*. 2001, pp. 159–166 (cit. on p. 34).

[TSH01b] X. Tricoche, G. Scheuermann, and H. Hagen. “Topology-based visualization of time-dependent 2D vector fields”. In: *Data Visualization 2001*. Springer, 2001, pp. 117–126 (cit. on p. 35).

[TSH20] X. Tricoche, W. Schlei, and K. C. Howell. “Extraction and visualization of Poincaré map topology for spacecraft trajectory design”. *IEEE Transactions on Visualization and Computer Graphics* 27:2, 2020, pp. 765–774 (cit. on p. 32).

[TWHS03] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. “Saddle connectors – an approach to visualizing the topological skeleton of complex 3D vector fields”. In: *Proc. IEEE Visualization*. 2003, pp. 225–232 (cit. on pp. 33, 75, 83, 94, 117, 146).

[TWHS04a] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. “Stream line and path line oriented topology for 2D time-dependent vector fields”. In: *Proc. IEEE Visualization*. 2004, pp. 321–238 (cit. on pp. 35, 97).

[TWHS04b] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. “Grid-independent detection of closed stream lines in 2D vector fields.” In: *Proc. Vision, Modeling and Visualization*. Vol. 4. 2004, pp. 421–428 (cit. on p. 31).

[TWHS05] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. “Topological methods for 2D time-dependent vector fields based on stream lines and path lines”. *IEEE Transactions on Visualization and Computer Graphics* 11:4, 2005, pp. 383–394 (cit. on p. 35).

[TWSH02] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen. “Topology tracking for the visualization of time-dependent two-dimensional flows”. *Computers & Graphics* 26:2, 2002, pp. 249–257 (cit. on p. 35).

[Üff+12] M. Üffinger, F. Sadlo, M. Kirby, C. Hansen, and T. Ertl. "FTLE computation beyond first-order approximation". In: *Short Paper Proc. Eurographics*. 2012, pp. 61–64 (cit. on p. 36).

[ÜSE13] M. Üffinger, F. Sadlo, and T. Ertl. "A time-dependent vector field topology based on streak surfaces". *IEEE Transactions on Visualization and Computer Graphics* 19:3, 2013, pp. 379–392 (cit. on pp. 38, 97, 102, 106, 119, 120).

[VP09] A. Van Gelder and A. Pang. "Using PVsolve to analyze and locate positions of parallel vectors". *IEEE Transactions on Visualization and Computer Graphics* 15:4, 2009, pp. 682–695 (cit. on p. 18).

[VPV+09] L. Van Der Maaten, E. Postma, J. Van den Herik, et al. "Dimensionality reduction: a comparative". *Journal of Machine Learning Research* 10:66-71, 2009, p. 13 (cit. on p. 41).

[Wan+13] W. Wang, X. Yan, C. Fu, A. Hanson, and P. Heng. "Interactive exploration of 4D geometry with volumetric halos". In: *Short Paper Proc. Pacific Conference on Computer Graphics and Applications*. 2013, pp. 1–6 (cit. on pp. 41, 77).

[WB96] C. Weigle and D. C. Banks. "Complex-valued contour meshing". In: *Proc. IEEE Visualization*. 1996, pp. 173–180 (cit. on p. 41).

[Wei+05] T. Weinkauf, H. Theisel, K. Shi, H.-C. Hege, and H.-P. Seidel. "Extracting higher order critical points and topological simplification of 3D vector fields". In: *Proc. IEEE Visualization*. 2005, pp. 559–566 (cit. on p. 35).

[Wei08] T. Weinkauf. "Extraction of Topological Structures in 2D and 3D Vector Fields". PhD thesis. University Magdeburg, 2008 (cit. on p. 74).

[WG09] T. Weinkauf and D. Günther. "Separatrix persistence: extraction of salient edges on surfaces using topological methods". *Computer Graphics Forum* 28:5, 2009, pp. 1519–1528 (cit. on p. 7).

[WG20] R. Witschi and T. Günther. "Implicit ray casting of the parallel vectors operator". In: *Short Paper Proc. IEEE Visualization*. 2020, pp. 31–35 (cit. on p. 18).

[WGS07] A. Wiebel, C. Garth, and G. Scheuermann. "Computation of localized flow for steady and unsteady vector fields and its applications". *IEEE Transactions on Visualization and Computer Graphics* 13:4, 2007, pp. 641–651 (cit. on p. 35).

[WHT12] T. Weinkauf, H.-C. Hege, and H. Theisel. “Advected tangent curves: a general scheme for characteristic curves of flow fields”. *Computer Graphics Forum* 31:2, 2012, pp. 825–834 (cit. on pp. 16, 37).

[Wie+07] A. Wiebel, X. Tricoche, D. Schneider, H. Jaenicke, and G. Scheuermann. “Generalized streak lines: analysis and visualization of boundary induced vortices”. *IEEE Transactions on Visualization and Computer Graphics* 13:6, 2007, pp. 1735–1742 (cit. on p. 15).

[Wie+11] A. Wiebel, R. Chan, C. Wolf, A. Robitzki, A. Stevens, and G. Scheuermann. “Topological flow structures in a mathematical model for rotation-mediated cell aggregation”. In: *Topological Methods in Data Analysis and Visualization*. Springer, 2011, pp. 193–204 (cit. on p. 121).

[Wig13] S. Wiggins. *Chaotic Transport in Dynamical Systems*. Vol. 2. Springer Science & Business Media, 2013 (cit. on p. 34).

[Wig90] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Vol. 2. Springer, 1990 (cit. on pp. 8, 32, 33).

[Wij93] J. J. van Wijk. “Implicit stream surfaces”. In: *Proc. IEEE Conference on Visualization*. 1993, pp. 245–252 (cit. on p. 41).

[WLG97] R. Wegenkittl, H. Löffelmann, and E. Gröller. “Visualizing the behavior of higher dimensional dynamical systems”. In: *Proc. IEEE Visualization*. 1997, 119–ff (cit. on p. 41).

[WRT18a] T. Wilde, C. Rössl, and H. Theisel. “FTLE ridge lines for long integration times”. In: *Proc. IEEE Visualization*. 2018, pp. 57–61 (cit. on p. 36).

[WRT18b] T. Wilde, C. Rössl, and H. Theisel. “Recirculation surfaces for flow visualization”. *IEEE Transactions on Visualization and Computer Graphics* 25:1, 2018, pp. 946–955 (cit. on pp. 32, 62, 63, 111, 121).

[WS01] T. Wischgoll and G. Scheuermann. “Detection and visualization of closed streamlines in planar flows”. *IEEE Transactions on Visualization and Computer Graphics* 7:2, 2001, pp. 165–172 (cit. on p. 31).

[WS02] T. Wischgoll and G. Scheuermann. “Locating closed streamlines in 3D vector fields”. In: *Proc. Symposium on Data Visualisation*. 2002, pp. 227–232 (cit. on pp. 31, 94).

[WSTH07] T. Weinkauf, J. Sahner, H. Theisel, and H.-C. Hege. “Cores of swirling particle motion in unsteady flows”. *IEEE Transactions on Visualization and Computer Graphics* 13:6, 2007, pp. 1759–1766 (cit. on pp. 25, 26, 64, 65, 110).

[WT10] T. Weinkauf and H. Theisel. “Streak lines as tangent curves of a derived vector field”. *IEEE Transactions on Visualization and Computer Graphics* 16:6, 2010, pp. 1225–1234 (cit. on pp. 15, 37, 121).

[WTHS04] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. “Boundary switch connectors for topological visualization of complex 3D vector fields”. In: *Proc. Eurographics / IEEE VGTC Symposium on Visualization*. 2004, pp. 183–192 (cit. on pp. 34, 94, 146).

[WTHS06] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. “Topological structures in two-parameter-dependent 2D vector fields”. *Computer Graphics Forum* 25:3, 2006, pp. 607–616 (cit. on pp. 21, 71).

[WTVP11] T. Weinkauf, H. Theisel, A. Van Gelder, and A. Pang. “Stable feature flow fields”. *IEEE Transactions on Visualization and Computer Graphics* 17:6, 2011, pp. 770–780 (cit. on pp. 21, 67).

[WWRT21] S. Wolligandt, T. Wilde, C. Rössl, and H. Theisel. “A modified double gyre with ground truth hyperbolic trajectories for flow visualization”. *Computer Graphics Forum*, 2021, pp. 209–221 (cit. on p. 39).

[Yan+21] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. “Scalar field comparison with topological descriptors: properties and applications for scientific visualization”. *Computer Graphics Forum* 40:3, 2021, pp. 599–633 (cit. on p. 7).

[ZP04] X. Zheng and A. Pang. “Topological lines in 3D tensor fields”. In: *Proc. IEEE Visualization*. 2004, pp. 313–320 (cit. on p. 20).