EXTRACTION OF TOPOLOGICAL STRUCTURES IN 2D AND 3D VECTOR FIELDS

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von Dipl.-Inf. Tino Weinkauf
geboren am 31. Oktober 1974 in Rostock

Gutachter:
Prof. Dr. Holger Theisel
Prof. Dr. Thomas Ertl
Prof. Alex Pang, Ph.D.

Magdeburg, den 07. März 2008

Danksagung

Wenn zwei Menschen über eine lange Zeit an verschiedenen Orten leben und sowohl ihre Freundschaft erhalten als auch erfolgreich gemeinsam arbeiten wollen, dann müssen sie sich blind verstehen. Mit meinem Freund und Doktorvater Holger Theisel kann ich das. Die letzten Jahre haben nicht einfach nur Spaß gemacht, sondern sie haben mich definiert. Deine Art zu forschen, nehme ich mir als Vorbild.

Meinem Gutachter Thomas Ertl möchte ich herzlich danken für sein tiefgehendes und sorgfältiges Gutachten dieser Arbeit.

Alex Pang reviewed this thesis and I am deeply grateful for this. It was great that you could make it to the disputation.

Meinem Freund und Kollegen Jan Sahner bin ich sehr dankbar für die wunderbare Zusammenarbeit bei der Forschung und beim Schreiben von Anträgen. Deine positive Grundeinstellung hat unsere Arbeit geprägt und alles ein bisschen leichter gemacht, selbst die quälenden Dinge.

Hans-Christian Hege gilt mein besonderer Dank. Seine Unterstützung über all die Jahre meiner Arbeit am ZIB ist einzigartig. Die Arbeitsatmosphäre am ZIB ist großartig und alle Kollegen sind mir sehr ans Herz gewachsen. Das hat sich nicht zuletzt auch positiv auf diese Arbeit ausgewirkt. Insbesondere möchte ich meinen Studenten Jan Reininghaus, Nathalie Teuber und Norbert Lindow für ihre Unterstützung danken.

Man kann eine aussagekräftige Visualisierung nur dann produzieren, wenn man den Hintergrund versteht. Bernd R. Noack hat sich viel Zeit genommen, um mir dieses Verständnis in vielen langen und schönen Gesprächen zu vermitteln. Bernd, ich danke Dir. Desweiteren gilt mein Dank auch den anderen Mitarbeitern des Instituts für Strömungsmechanik und Technische Akustik der TU Berlin, die mir immer hilfreich zur Seite standen.

Meine Eltern haben mich immer auf meinem Weg begleitet. Eure Energie hat mir viel Kraft gegeben, eure Geduld die Ausdauer. Ich danke Euch.

Mit wenigen Menschen schwingt man im Takt. Mit Dir, Wenke, ist es eine Sinfonie.

Die Arbeit an meiner Dissertation hat viel Kraft und Zeit erfordert, und ich bin sehr glücklich darüber, dieses aufbringen zu können. Mit Eurer Hilfe.

Tino Weinkauf
23. April 2008

Abstract

Analyzing large and high-dimensional flow data sets is a non-trivial task and favorably carried out using sophisticated tools which allow to concentrate on the most relevant information and to automate the analysis. These goals can be achieved using topological methods, which foster target-oriented studies of the most important flow features. They have a variety of applications which ranges from a skeletal representation of the overall flow behavior to a detailed analysis of vortex structures.

This thesis presents novel algorithms and approaches for the extraction, tracking and visualization of topological structures of vector fields. The new concept of connectors is introduced which allows visually simplified representations of topological skeletons of complex 3D vector fields. The first visualization technique for 3D higher order critical points and the corresponding classification are presented. Based on this theory, two novel applications for the topological simplification and construction of 3D vector fields have been developed. Furthermore, the first generic approach to feature extraction is presented, which allows to extract and track a rich variety of geometrically defined, local and global features evolving in scalar and vector fields. The use of generic concepts and grid independent algorithms aims at a broad applicability of the extraction methods while alleviating the implementational expenses. Further contributions include the first topology-based visualization approach for two-parameter-dependent 2D vector fields and a thorough study of vortex structures. The usefulness of the newly developed methods is shown by applying them to analyze a number of data sets.

The work presented in this thesis has been published in peer-reviewed international conference proceedings, journals, and books.

Zusammenfassung

Die Analyse von großen und hochdimensionalen Strömungsdaten stellt eine Herausforderung dar, bei der sich die Verwendung von automatisierbaren Werkzeugen bewährt hat, die eine Konzentration auf die wesentlichen Informationen erlauben. Topologische Analyseverfahren erlauben eine zielorientierte Untersuchung der relevantesten Strömungsmerkmale und lassen sich vielfältig anwenden, von der skelettartigen Darstellung des Strömungsverhaltens bis hin zur detaillierten Analyse von Wirbelstrukturen.

Diese Dissertation stellt neue Algorithmen und Ansätze zur Extraktion, zeitlichen Verfolgung und Visualisierung von topologischen Strukturen in Vektorfeldern vor. Zur vereinfachten Darstellung topologischer Skelette von komplexen 3D Vektorfeldern wurde das neue Konzept der Konnektoren entwickelt. Die erste Visualisierungstechnik für 3D kritische Punkte höherer Ordnung und die zugehörige Klassifikation werden vorgestellt. Auf Basis dieser Theorie wurden zwei neue Anwendungen zur topologischen Simplifizierung und Konstruktion von 3D Vektorfeldern entwickelt. Desweiteren wird der erste generische Ansatz zur Extraktion von Merkmalen vorgestellt, der es erlaubt, eine Vielzahl von geometrisch definierten, lokalen und globalen Merkmalen in Skalar- und Vektorfeldern zu extrahieren und zeitlich zu verfolgen. Die Verwendung generischer Konzepte und gitterunabhängiger Algorithmen zielt auf eine breite Anwendbarkeit der Methoden bei gleichzeitig vergleichsweise geringem Implementationsaufwand. Zu den weiteren Beiträgen der Arbeit gehören der erste topologiebasierte Ansatz zur Visualisierung von zwei-parameterabhängigen 2D Vektorfeldern und eine umfassende Studie von Wirbelstrukturen. Die Nützlichkeit der Methoden wird anhand der Analyse verschiedener Datensätze gezeigt.

Die in dieser Dissertation vorgestellten Arbeiten wurden in begutachteten internationalen Konferenzbänden, Zeitschriften und Büchern veröffentlicht.

Contents

Chapter 1

Introduction

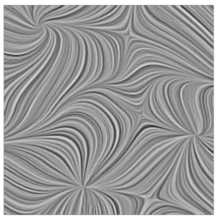
Computing power increases constantly. While the fastest supercomputer in 1993 had a performance of 59.7 GFlop/s, the fastest installation in 2007 reached 280.6 TFlop/s [MSDS]. This is an increase by a factor of 4700. As of this writing, the petascale era is already approaching. Along with the computing power, the size and complexity of simulation results is increasing as well. In many cases the simulated data sets are at least four-dimensional, e.g. with three spatial dimensions and time. In some cases even higher-dimensional data sets are produced by considering additional parameters.

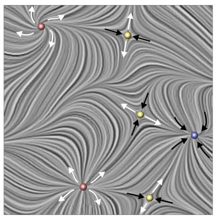
Due to the sheer size of the data sets alone, it is favorable if not necessary to automate at least parts of the analysis. A way to achieve this is by extracting features. A *feature* – as used in this thesis – is a mathematically well-defined, geometric object (point, line, surface, ...) with its definition and interpretation depending on the underlying application, but usually it represents important structures (e.g. vortex, stagnation point) or changes to such structures (events, bifurcations). An automated extraction of features aids an analysis in a number of ways:

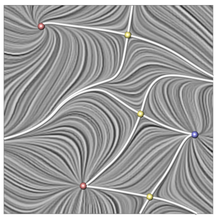
- **reduction of information**
  The human brain has the ability to grasp primarily three dimensions and the current hardware has only limited capabilities of displaying them. Hence, only parts of the massive and complex simulation results can be visualized at once. Feature extraction reduces the amount of data to a small set of geometric objects. Furthermore, a quantification of the extracted features allows to build up a feature hierarchy leading to even further simplified representations by e.g. filtering out the less important features.

- **target-oriented study**
  Feature extraction is used to automatically find interesting parts in the data, where e.g. certain structural changes occur. This can guide the user in the manual exploration of a data set. It allows to concentrate on certain aspects of the data – leading to a target-oriented, application-dependent study of the most important structures of a data set.

- **shifting the analysis to the supercomputer**
  In some cases it is preferable to analyze the data on the supercomputer along the simulation: for example, if the data set is too large to be efficiently handled by commodity hardware, or if the analysis results have some influence on the


(a) Stream lines.


(b) Sources (red), sinks (blue), and saddles (yellow).


(c) Separation lines emanating from saddles.

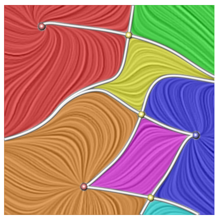

(d) Sectors of different flow behavior.

Figure 1.1: Topology of a simple 2D vector field.

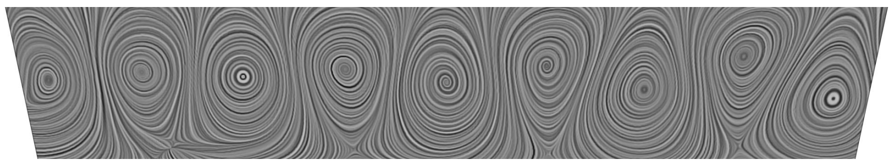
simulation itself (e.g. simulation steering). Feature extraction is easily automated and therefore, it is perfectly fitted for batch jobs on supercomputers.

- **interactive visualization**
  The resulting feature set is usually small enough to be handled and displayed interactively with commodity hardware.
- **more objective analysis**
  In contrast to most other visualization methods, feature extraction techniques usually depend on less parameters or even no parameters at all. Hence, the interpretation of the results depends less on a user-defined parametrization (e.g. isovalue, transfer function).
- **faster analysis**
  The time needed by the user is reduced since parts of the analysis are automated and the manual part is interactive.

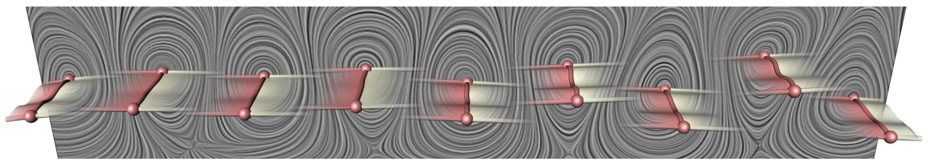
This thesis is concerned with the feature-based analysis of vector fields, with the main focus on flow fields which play a vital role in many areas. Examples are burning chambers, turbomachinery and aircraft design in industry as well as blood flow in medicine.

The analysis is based on the extraction and examination of topological structures of vector fields. Topology is a well-researched field of mathematics. It allows to condense a data set to its structural skeleton. For vector fields, this means to segment the domain into regions of different flow behavior. Consider a simple 2D vector field as shown in figure 1.1a. A topological analysis always starts with the extraction of so-called critical points which are the zeros of the vector field. As it can be observed in figure 1.1b there are different flow patterns around critical points which allow to classify them into sources, sinks, and saddles. While the flow behavior around sources and sinks is uniformly either outflow or inflow, the flow around saddles is a mixture of both. Certain stream lines around saddles can be found which separate these different areas. They are called separation lines (figure 1.1c). This leads to a complete segmentation of the domain into regions of different flow behavior as highlighted in figure 1.1d.

Topology can be used in a number of ways to foster the analysis of flows. For example, the topology of the velocity field of a flow can be seen as a condensed representation of the stream lines and may therefore serve as a skeletal, simplified representation of the flow. However, such an analysis depends on the reference frame and may therefore be not applicable in all situations. As another application, the topological



(a) The line integral convolution method depicts the vortices, but is incapable of measuring properties like their velocity.



(b) A feature-based analysis extracts the cores of the vortices as distinct geometric objects (lines) and tracks them over time (surfaces).

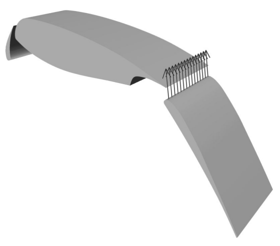
Figure 1.2: Difference between visual and feature-based analysis at a flow behind a cylinder. Data courtesy of Bernd R. Noack (TU Berlin) and Gerd Mutschke (FZ Rossendorf).

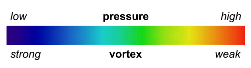
structures of certain derived vector fields describe centers of high strain or vortex activity – leading to a Galilean invariant analysis of important flow processes like mixing.

In the following we give two examples of how flows can be analyzed using the theory and tools developed in this thesis.

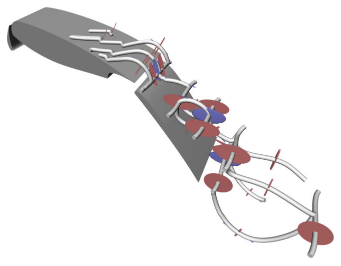
A feature-based analysis gives rise to new possibilities in comparison to other visualization techniques. Consider the flow behind a cylinder as shown in figure 1.2. The so-called von Kármán vortex street develops behind the cylinder and is clearly depicted by the line integral convolution method shown in figure 1.2a. Using an animation, the temporal movement of these vortices can be depicted by this method as well. However, it is incapable of *distinguishing* between different vortices and *measuring* their path, velocity, or life time. The feature-based vortex analysis shown in figure 1.2b allows this since the vortices have been extracted as distinct geometric objects. Furthermore, it is possible to quantify these objects by means of importance or strength, and to filter accordingly.

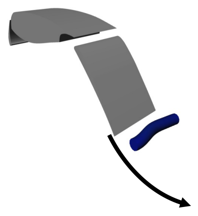
Figure 1.3 shows an example where a topology-based analysis has been successfully applied to explain the impact of an active flow control technique at an airfoil. The flow around an airfoil is subject to large efforts in order to increase the desired lift and to reduce the parasitic drag. In this example, these performance enhancements are achieved by controlling the flow separation at the rear flap using periodic air injection (figure 1.3a). The uncovering of the underlying physics was a necessary step in order to choose optimal values for frequency, intensity, and angle of the injection. Based on this, the lift could be raised by 11.2%. The vortex structures have been extracted as topological separatrices of the pressure gradient and denote lines of minimal pressure. Figure 1.3c shows parts of the topological skeleton of the pressure gradient. A quantification of the separation lines based on pressure and a subsequent filtering of weak vortices has been applied. The result is shown in figures 1.3d–f where the impact of the frequency of air injection onto the vortex structures can be studied. Note that this is a five-dimensional data set consisting of three spatial dimensions, time, and the parameter dimension. Raising the frequency causes a reduction of the lower vortex, which

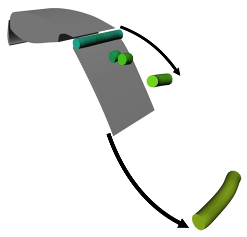

(a) Periodic excitation by suction and blowing at the rear flap.



(b) Colormap used to indicate flow pressure and vortex strength.


(c) A subset of the topological skeleton of the pressure gradient denotes minimal pressure, i.e., vortex cores. $F^+ = 2.0$.


(d) Vortex structures of the unexcited flow.


(e) Vortex structures of the optimally excited flow ($F^+ = 0.6$). Gain of lift: 11.2%.

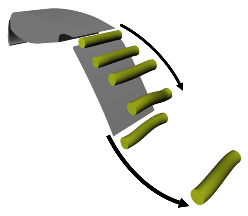

(f) Vortex structures of the high-frequency excited flow ($F^+ = 2.0$). Gain of lift: 6.1%.

Figure 1.3: A topology-based vortex analysis of the flow around an airfoil elucidates the impact of an active flow control technique and explains why a high-frequency excitation leads to a smaller gain of lift. Data courtesy of Bert Günther (TU Berlin).

is a necessary condition for gaining lift. However, higher frequencies ($F^+ > 0.6$) are not beneficial to the lift. Using a visual comparison of the vortex structures at different frequencies, we found that new vortex structures are induced by the air injection itself. This has a negative effect on the pressure ratio and consequently on the lift. Especially at higher frequencies, the excitation dominates the natural flow structures and induces long-living, almost two-dimensional vortices in fast succession at the top of the rear flap (figure 1.3f). In contrast to this, the induced vortices at $F^+ = 0.6$ dissolve quickly and therefore, they are less influential. Our topology-based analysis technique of the vortex structures contributed to the physical understanding of the flow structures and was a substantial part of the optimal choice of parameters.

In chapter 2 we discuss the theoretical background needed for this thesis and add the notions of saddle connectors and boundary switch connectors to the body of theory. Furthermore, a geometrical description of the flow behavior around higher order critical points is introduced.

The main part of this thesis, chapters 3 – 6, deals with the extraction of topological structures. The foundation of most extraction techniques developed in the course of this thesis is presented in chapter 3: the so-called Unified Feature Extraction Architecture defines a small but comprehensive set of algorithms and concepts, which allows to extract and track a variety of topological and other features. It is a major contribution of this thesis and the first generic approach to feature extraction treating local and global features.

Based on this architecture we formulate most of the extraction techniques in the following chapters, which deal with first order topological structures of steady vector fields (chapter 4), time-dependent and two-parameter-dependent vector fields (chapter 5), or higher order topological structures (chapter 6). Major contributions in these chapters are the extraction of connectors, several novel algorithms for the treatment of global bifurcations, the first topology-based visualization approach for two-parameter-dependent vector fields, and the first topological simplification method for 3D vector fields. Along with the development of these methods we show their usefulness by applying them to analyze a number of data sets.

Further applications of topological concepts and methods are presented in chapter 7. We show how to construct 3D vector fields from a given topological description, or how the results of a topological analysis can be used to steer other visualization techniques. The main part of this chapter, however, is devoted to the treatment of vortex core lines. We show how these lines can be extracted and tracked using the methods developed in previous chapters. Furthermore, we give a unified notation of cores of swirling motion, present a novel way of describing vortex core lines as extremum lines, and link the fields of topology and vortex analysis by showing how vortex core lines can be described as topological separatrices of a derived vector field.

We discuss the applicability of topological methods in chapter 8. Among other things, we comment on noise sensitivity and turbulent flows, and we study the topological complexity of 2D and 3D vector fields. Chapter 9 draws conclusions and gives an outlook to future work.

The work presented in this thesis has been published in peer-reviewed international conference proceedings, journals, and books:

- Extracting and tracking topological structures: [TWHS03, WTHS04a, TWHS04a, TWHS04b, TWHS05, WTHS06]
- Simplification and construction: [WTHS04b, WTS+05]
- Concepts and theoretical foundations: [WTHS07, TWHS07, WST+07, TRW08]
- Vortex analysis: [SWH05a, TSW+05, SWTH07, WSTH07]
- Applications: [WHN+02, WHN+03, HWPH04, WNC+04, HWPH05, PWS+06, GTP+07]

1.1 Related Work

In the following we give an overview of related work. We restrict the discussion to the field of flow visualization with a major focus on topology-based analysis. Following the approach of Post et al. [PVH+02], we subdivide the field of flow visualization into *direct*, *integration-based*, and *feature-based* methods.

Direct flow visualization methods map the data to an image without involved computations. Examples are color coding [SRBE99], volume rendering [CMBC93, EYSK94, RLMO03], arrow plots [KH91, BP96], or combinations thereof [KML99].

The class of integration-based methods can be subdivided into *texture-based* and *geometry-based* techniques. All these methods employ numerical integration schemes which introduce a certain amount of uncertainty due to integration errors. Lodha et al. [LPSW96] and Pang et al. [PWL97] provide a number of solutions for depicting this uncertainty.

Van Wijk introduced the first texture-based method called spot noise [vW91], which has later been extended to include magnitude information [dLvW95]. Cabral and Leedom [CL93] developed the line integral convolution method (LIC), where a noise texture is filtered along the stream lines of the vector field. This yields dense representations. Stalling and Hege [SH95a] significantly lowered the computation times for LIC by exploiting coherence along the stream lines. Later improvements include directional clues [WGP97, WLG97], animation [SK97], GPU-accelerated implementations [HWSE99], extension to surfaces [BSH96, SH97, MKFI97] and to 3D [IG97, RSHTE99]. A comparison between LIC and spot noise can be found in [dLvL98].

Some work has been done on the field of texture advection for the visualization of time-dependent vector fields: Moving Textures by Max and Becker [MB95], the Lagrangian-Eulerian Advection method (LEA) by Jobard et al. [JEH02], and Image Based Flow Visualization (IBFV) by van Wijk [vW02] to name a few. The latter has been extended to 3D surfaces [vW03, LJH03] and volumes [TvW03]. Weiskopf et al. [WEE03] present a generic framework for texture-based visualization of 2D unsteady flows. A thorough overview of texture-based methods can be found in [LHD$^+$04].

Geometry-based methods visualize the characteristic curves of flow fields (see also section 2.1.2) or parts thereof as points, lines, surfaces, or volumes. Kenwright and Lane [KL96] present a particle tracer for time-dependent flows which gains its interactivity from tetrahedral decomposition. Two approaches to accelerate particle tracing on sparse grids are given by Teitzel and Ertl [TE99]. Krüger et al. [KKKW05] present a highly interactive particle system implemented on the GPU. Line-type structures such as stream or path lines can be visualized using the illumination method of Zöckler et al. [ZSH96], which allows a fast cylindrical shading using a texture lookup. Mallo et al. [MPSS05] improved the method by incorporating a different diffuse reflection term. A number of methods have been proposed to distribute the seeding points [ZSH96, VKP00, YKP05], or to distribute the stream lines themselves [TB96, JL97, MAD05]. The computation of stream surfaces can be reduced to the integration of a finite number of adjacent stream lines and appropriate triangulation between them [Hul92, Sta98, GTS$^+$04a]. Implicit stream surfaces are computed by [vW93]. Krauskopf et al. [KOD$^+$05] give an overview of different methods for computing stream surfaces. Löffelmann et al. [LMG97] propose the concept of stream arrows to enhance the visualization of stream surfaces by cutting away arrow-like areas encoding flow direction or divergence. Similarly to stream surfaces, stream volumes can be computed by a proper tetrahedrization between adjacent stream lines as proposed by Max et al. [MBC93]. An appropriate projection of the tetrahedra yields smoke-like images. An extension to unsteady flows is given by Becker et al. [BML95].

A number of further geometry-based approaches have been derived from the standard set of stream objects to encode additional information: examples are stream ribbons, streamtubes, and streamballs, see [PVH$^+$02].

Verma et al. [VKP99] presented a method to compare texture-based and geometry-based methods by mapping between a LIC and a stream line visualization – allowing a smooth transition.

Feature-based methods aim at presenting the most important parts of a data set at a high level of abstraction. The definition of importance depends on the application.

Among the features of interest are topological structures, vortices, as well as other features such as attachment and detachment lines or shock waves.

Topology in general is well-understood and thorough treatments can be found in [AS92, Bak91, GH86, Asi93, FG82].

Topological methods have been introduced as a visualization tool by Helman and Hesselink in their seminal work [HH89], where first order critical points are classified by an eigenvalue/eigenvector analysis of the Jacobian matrix, and separatrices starting from saddle points and from attachment and detachment points at no-slip boundaries are considered. This work has later been extended to 3D by the same authors [HH91]. Globus et al. [GLL91] present a system for visualizing the topological skeleton of 3D vector fields using icons and separation curves.

Since then a considerable amount of research has been done to extract, analyze, modify and visualize the topology of vector fields.

Several approaches can be used to extract critical points. In piecewise linear fields, the zeros can be computed explicitly. In more general settings, one might use a Newton-Raphson approach. An octree-like method is presented by Mann et al. [MR02]: they compute the index of each cell and a non-zero index triggers a recursive subdivision. An approach specifically designed for electric fields defined by a set of point charges has been presented by Max et al. [MW07]. Trotts et al. [TKH00] introduce the notion of critical points at infinity to find new separatrices. The curvature of stream lines in the proximity of critical points has been studied by Theisel and Weinkauf [The95, WT02] for 2D and 3D vector fields.

Regions of different flow behavior on the boundary of 2D vector fields as well as the corresponding separatrices have been considered by de Leeuw and van Liere [dLvL99a] and Scheuermann et al. [SHJK00].

Another type of separatrices are closed stream lines. A first approach to detecting closed stream lines was given by Wischgoll et al. [WS01] which uses the underlying grid structure of a piecewise linear vector field: each grid cell is analyzed concerning the re-entering behavior of the stream lines starting at its boundaries. Closed stream lines in 3D vector fields are discussed in [Asi93, PS07].

A thorough study of higher order critical points, i.e., critical points with a possibly vanishing Jacobian, in 2D vector fields has been given by Firby and Gardiner [FG82]. Li et al. [LVRL06] discuss how to represent these points on triangular surfaces using a carefully chosen triangulation and interpolation. Scheuermann et al. [SHK$^+$97, SKMR98] give visualization approaches for planar flows.

For parameter-dependent vector fields, one aims at capturing the evolution of the topological structures. An important class of such fields are time-dependent flows. A common way of tracking the evolution is to extract the features in each time step and establish a correspondence afterwards based on Euclidean distance and feature attributes.

Tricoche et al. [TSH01b, TWSH02] track the location of critical points over time and detect local bifurcations like fold bifurcations and Hopf bifurcations. This approach works on a piecewise linear 2D vector field and computes and connects the critical points on the faces of a prism cell structure, which is constructed from the underlying triangular grid. An extension to 3D has been given by Garth et al. [GTS04b] together with a novel plot-like depiction of the critical points. Wischgoll et al. [WSH01] track closed stream lines over time by applying a contouring&connecting-like approach: at each time step closed stream lines are detected independently of each other, then the corresponding lines in adjacent time steps are connected.

A grid-independent solution to feature tracking has been given by Theisel and Seidel [TS03]. The basic idea is to track features of a time-dependent vector field by means of a stream line integration in a derived vector field – the so-called Feature Flow Field. This has been applied in [TS03] to track critical points. A number of further applications have been developed since then, see section 3.1 for an overview. A comparable approach to tracking critical points in scale space is given by Klein and Ertl [KE07].

Some work has been done to measure the distance of vector fields by means of topology. This generally means to detect and match the critical points of two vector fields: for each critical point in the first vector field a corresponding critical point in the second vector field has to be found, and vice versa. Then the distances between all corresponding critical points are compared: their summation gives the distance of the two vector fields. This way the computation of the distance of two vector fields is reduced to the computation of the distance of critical points. Theisel et al. [TW02] introduce the (γ, r) phase plane which maps all first order critical points of 2D vector fields to the area of the unit circle. Based on this, a corresponding metric is introduced which defines the distance between critical points as their euclidean distance in the (γ, r) phase plane. Based on this and the Feature Flow Field approach, Theisel et al. [TRS03c] propose a comparison technique for 2D vector fields. Depardon et al. [DLBB07] extended this approach and applied it to identify periodic phenomena from insufficiently time-resolved data sets measured using PIV. Previously, a different approach has been proposed by Lavin et al. [LBH98]. Here, the so-called (α, β) phase plane maps the critical points onto the unit circle only. Batra et al. [BKH99] give an extension by incorporating the connectivity of critical points. Furthermore, an extension to 3D is given [BH99].

Simplification of topological structures becomes important for complex vector fields in order to create expressive visualizations. The general idea is to build up a feature hierarchy and depict only the features above a certain user-defined threshold. Such a method is presented by de Leeuw and van Liere [dLvL99a, dLvL99b]. Tricoche et al. [TSH00] utilize the theory of higher order critical points to replace a cluster of first order 2D critical points with an higher order icon. An approach to a continuous topology simplification of 2D vector fields is presented by Tricoche et al. in [TSH01a]. Based on the graph structure of the topological skeleton and certain relevance measures, pairs of critical points are removed by local changes to the vector values at the grid nodes.

A number of other vector field modifications have been proposed which take care of the underlying topology. Westermann et al. [WJE01] propose a topology-preserving smoothing method. Lodha et al. [LRR00, LFR03] give different approaches to compressing vector fields. Theisel et al. [TRS03b] proposes a compression method based on topological construction, and another one combining compression and simplification [TRS03a]. Theisel et al. [The02] presents a method for designing 2D vector fields based on a topological description. Chen et al. [CML$^+$07] use Morse decomposition to edit the topology of vector fields on surfaces.

Topology has been used in a number of applications to study certain phenomena. Sun et al. [SBSH04] relate the positions of critical points and their connectivity to the efficiency of optical transmission through certain nano-scaled apertures. Hauser et al. [LDG98, HG00] extract and analyze the topological skeletons of particular analytic 3D vector fields. Tricoche et al. [TGK$^+$04] and Garth et al. [GTS04b] analyze the breakdown of vortices using topology. Garth et al. [GLT$^+$07] study swirl and tumble motion from engine simulation data. Different visualization methods including topology have been applied by Laramee et al. [LGD$^+$05] to study the flow through a cooling jacket. The topological visualization of this flow has been compared with an interactive, feature-based approach by Hauser et al. [HLD07].

Other types of features are considered as well. While we defer the discussion of vortices to section 7.2, we give a brief overview of other feature-based methods here.

Shock waves are important structures in flows around aircrafts. Their impact on the aircraft may cause structural instability and has therefore to be studied well. Lovely and Haimes give an algorithm based on isosurface extraction of a certain derived quantity. Pagendarm et al. [PS93] and Ma et al. [MRV96] propose methods based on the extraction of maxima of the density gradient. Haller [Hal04] and Surana et al. [SGH06] give a profound analysis of flow separation in 2D and 3D flows. Further algorithms for treating attachment and detachment lines are given by Kenwright et al. [KHL99]. Recirculation zones have been extracted by Haimes [Hai99]. Further techniques for tracking features are given in [SX97, RPS99, RPS01, SSZC94, KS91].

Thorough overviews of the state of the art can be found in Post et al. [PVH+02] for the whole field of flow visualization, in Laramee et al. [LHD+04] for texture-based techniques, in Post et al. [PVH+03] for feature-based methods, and in Laramee et al. [LHZP07] for topology-based methods.

Chapter 2

Theory

2.1 Notations and Definitions

In the following we introduce notations and definitions which will be used throughout this thesis.

2.1.1 Basic Notations

In this thesis we write scalars using small letters (s), points and vectors using bold small letters ($\mathbf{v}$) and matrices using bold capital letters ($\mathbf{T}$).

Let $\mathbb{E}^{n+p}$ be the $(n+p)$-dimensional Euclidean space equipped with a cartesian coordinate system and consisting of n spatial and p parameter dimensions. Furthermore, let $\mathbb{R}^{m\times b}$ be a $(m \times b)$-dimensional vector space. Henceforth, a *n-dimensional field depending on p parameters* is a map

$$\phi : \mathbb{E}^{n+p} \to \mathbb{R}^{m\times b} \quad \text{with} \quad n,m,b > 0;\ p \geq 0 \tag{2.1}$$

Depending on p we distinguish between

- $p = 0$: *steady* or *purely spatial* or *parameter-independent* fields,
- $p = 1$: *one-parameter dependent* fields (e.g. time-dependent fields),
- $p = 2$: *two-parameter dependent* fields (e.g. fields depending on time and a scale-space parameter).

A *scalar field* is given by $m \times b = 1$ and shall be written

$$s(\mathbf{x},\mathbf{a}) \in \mathbb{R} \quad \text{with} \quad \mathbf{x} \in \mathbb{E}^n;\ \mathbf{a} \in \mathbb{E}^p \tag{2.2}$$

where $\mathbf{x}$ is a spatial location and $\mathbf{a}$ represents a parameter combination. Purely spatial scalar fields are simply notated as $s(\mathbf{x})$.

A *vector field* is given by $m \times 1 > 1$. In the course of this thesis we will mainly deal with vector fields having as much components as spatial dimensions, i.e., $n = m > 1$. They shall be written as

$$\mathbf{v}(\mathbf{x},\mathbf{a}) = \begin{pmatrix} c_1(\mathbf{x},\mathbf{a}) \\ \vdots \\ c_n(\mathbf{x},\mathbf{a}) \end{pmatrix} \in \mathbb{R}^n \quad \text{with} \quad \mathbf{x} \in \mathbb{E}^n;\ \mathbf{a} \in \mathbb{E}^p. \tag{2.3}$$

Again, purely spatial vector fields are simply notated as $\mathbf{v}(\mathbf{x})$.

A *tensor field* is given by $m, b > 1$ and shall be written

$$\mathbf{T}(\mathbf{x},\mathbf{a}) = \begin{pmatrix} c_{11}(\mathbf{x},\mathbf{a}) & \dots & c_{1b}(\mathbf{x},\mathbf{a}) \\ \vdots & & \vdots \\ c_{m1}(\mathbf{x},\mathbf{a}) & \dots & c_{mb}(\mathbf{x},\mathbf{a}) \end{pmatrix} \in \mathbb{R}^{m\times b} \quad \text{with} \quad \mathbf{x} \in \mathbb{E}^n;\ \mathbf{a} \in \mathbb{E}^p. \tag{2.4}$$

Again, purely spatial tensor fields are simply notated as $\mathbf{T}(\mathbf{x})$. In the course of this thesis we will mainly deal with tensor fields being derivatives of scalar or vector fields.

A *2D vector field* shall be written as

$$\mathbf{v}(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix}. \tag{2.5}$$

Its first-order derivative is a 2×2 tensor field, called *Jacobian matrix*

$$\mathbf{J}(x,y) = \begin{pmatrix} \frac{d}{dx}u(x,y) & \frac{d}{dy}u(x,y) \\ \frac{d}{dx}v(x,y) & \frac{d}{dy}v(x,y) \end{pmatrix} = \begin{pmatrix} u_x(x,y) & u_y(x,y) \\ v_x(x,y) & v_y(x,y) \end{pmatrix}. \tag{2.6}$$

Similarly, a *3D vector field* shall be written as

$$\mathbf{v}(x,y,z) = \begin{pmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{pmatrix}. \tag{2.7}$$

Its first-order derivative is a 3×3 tensor field, called *Jacobian matrix* and written with a similar notation as in the 2D case

$$\mathbf{J}(x,y,z) = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix}. \tag{2.8}$$

Other directional derivatives as well as higher-order derivatives are indicated using the same index notation.

2.1.2 Characteristic Curves of Vector Fields

A curve $L \subset \mathbb{E}^n$ is called a *tangent curve* of a vector field $\mathbf{v}(\mathbf{x})$, if for all points $\mathbf{p} \in L$ the tangent vector of L coincides with $\mathbf{v}(\mathbf{p})$. Tangent curves are the solutions of the autonomous ODE system

$$\frac{d}{d\tau}\mathbf{x}(\tau) = \mathbf{v}(\mathbf{x}(\tau)) \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{2.9}$$

For all points $\mathbf{x} \in \mathbb{E}^n$ with $\mathbf{v}(\mathbf{x}) \neq \mathbf{0}$, there is one and only one tangent curve through it. Tangent curves do not intersect or join each other. Hence, tangent curves uniquely describe the directional information and are therefore an important tool for visualizing vector fields.

The tangent curves of a parameter-independent vector field $\mathbf{v}(\mathbf{x})$ are called *stream lines*. A stream line describes the path of a massless particle in $\mathbf{v}$.

In a one-parameter-dependent vector field $\mathbf{v}(\mathbf{x},t)$ there are four types of characteristic curves: stream lines, path lines, streak lines and time lines. To ease the explanation, we consider $\mathbf{v}(\mathbf{x},t)$ as a time-dependent vector field in the following: In a space-time point $(\mathbf{x}_0,t_0)$ we can start a *stream line* (staying in time slice $t=t_0$) by integrating

$$\frac{d}{d\tau}\mathbf{x}(\tau)=\mathbf{v}(\mathbf{x}(\tau),t_0) \quad \text{with} \quad \mathbf{x}(0)=\mathbf{x}_0 \tag{2.10}$$

or a *path line* by integrating

$$\frac{d}{dt}\mathbf{x}(t)=\mathbf{v}(\mathbf{x}(t),t) \quad \text{with} \quad \mathbf{x}(t_0)=\mathbf{x}_0. \tag{2.11}$$

Path lines describe the trajectories of massless particles in time-dependent vector fields. The ODE system (2.11) can be rewritten as an autonomous system at the expense of an increase in dimension by one, if time is included as an explicit state variable:

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}\\ t\end{pmatrix}=\begin{pmatrix}\mathbf{v}(\mathbf{x}(t),t)\\ 1\end{pmatrix} \quad \text{with} \quad \begin{pmatrix}\mathbf{x}\\ t\end{pmatrix}(0)=\begin{pmatrix}\mathbf{x}_0\\ t_0\end{pmatrix}. \tag{2.12}$$

In this formulation space and time are dealt with on equal footing – facilitating the analysis of spatio-temporal features. Path lines of the original vector field $\mathbf{v}$ in ordinary space now appear as tangent curves of the vector field

$$\mathbf{p}(\mathbf{x},t)=\begin{pmatrix}\mathbf{v}(\mathbf{x},t)\\ 1\end{pmatrix} \tag{2.13}$$

in space-time. To treat stream lines of $\mathbf{v}$, one may simply use

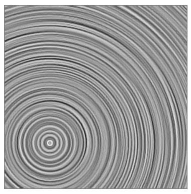
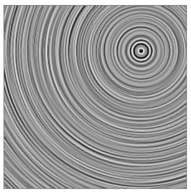
$$\mathbf{s}(\mathbf{x},t)=\begin{pmatrix}\mathbf{v}(\mathbf{x},t)\\ 0\end{pmatrix}. \tag{2.14}$$

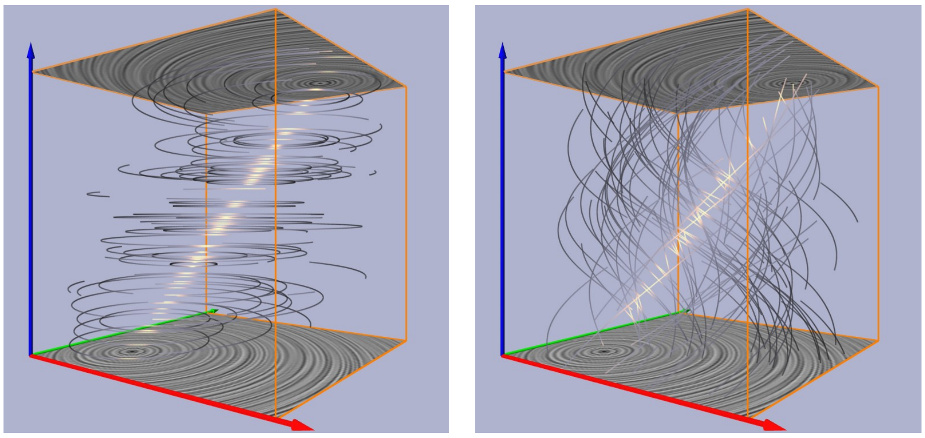
Figure 2.1 illustrates $\mathbf{s}$ and $\mathbf{p}$ for a simple example vector field $\mathbf{v}$. It is obtained by a linear interpolation over time of two bilinear vector fields.

A *streak line* is the connection of all particles set out at different times but the same point location. In an experiment, one can observe these structures by constantly releasing dye into the flow from a fixed position. The resulting streak line consists of all particles which have been at this fixed position sometime in the past. Considering the vector field $\mathbf{p}$ introduced above, streak lines can be obtained in the following way: apply a stream surface integration in $\mathbf{p}$ where the seeding curve is a straight line segment parallel to the t-axis, a streak line is the intersection of this stream surface with a hyperplane perpendicular to the t-axis (Figure 2.2a).

A *time line* is the connection of all particles set out at the same time but different locations, i.e., a line which gets advected by the flow. An analogon in the real world is a yarn or wire thrown into a river, which gets transported and deformed by the flow. However, in contrast to the yarn, a time line can get shorter and longer. It can be obtained by applying a stream surface integration in $\mathbf{p}$ starting at a line with $t=$ const., and intersecting it with a hyperplane perpendicular to the t-axis (Figure 2.2b).

Streak lines and time lines can not be described as tangent curves. Both types of lines fail to have a property of stream and path lines: they are not locally unique, i.e., for a particular location and time there is more than one streak and time line passing through. However, stream, path and streak lines coincide for steady vector fields $\mathbf{v}(\mathbf{x},t)=\mathbf{v}(\mathbf{x},t_0)$ and are described by (2.9) in this setting. Time lines do not fit into this.

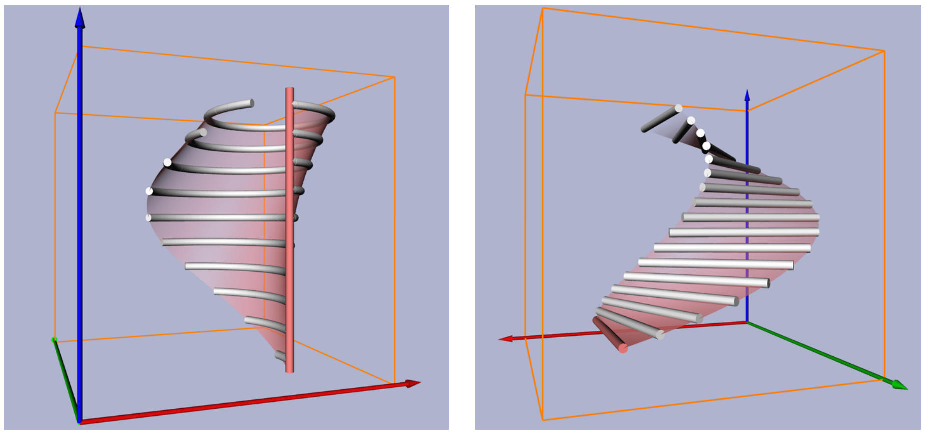
$\mathbf{v}(x,y,t) \;=\; (1-t)\cdot$  $+\,t\,\cdot$ 



(a) Tangent curves of **s** correspond to the stream lines in **v**. See (2.14).

(b) Tangent curves of **p** correspond to the path lines in **v**. See (2.13).

Figure 2.1: Characteristic curves of a simple 2D time-dependent vector field shown as illuminated field lines. The red/green coordinate axes denote the (x,y)-domain, the blue axis shows time.



(a) Streak lines of **v**.

(b) Time lines of **v**.

Figure 2.2: Characteristic curves of a simple 2D time-dependent vector field. Seeding curves and resulting stream surfaces are colored red. Same data set as in Figure 2.1.

For two-parameter-dependent vector fields $\mathbf{v}(\mathbf{x},s,t)$ we consider only stream lines for a given parameter tuple (s,t). Similar to (2.14) they are given as the tangent curves of

$$\mathbf{s}(\mathbf{x},s,t) = \begin{pmatrix} \mathbf{v}(\mathbf{x},s,t) \\ 0 \\ 0 \end{pmatrix}. \tag{2.15}$$

2.1.3 Derived Measures of Vector Fields

A number of measures can be derived from a vector field $\mathbf{v}$ and its derivatives. These measures indicate certain properties and can be helpful when visualizing flows.

The *magnitude* of $\mathbf{v}$ is given as

$$|\mathbf{v}| = \sqrt{u^2 + v^2 + w^2}. \tag{2.16}$$

The *divergence* of a flow field is given as

$$\operatorname{div}(\mathbf{v}) = \nabla \cdot \mathbf{v} = \operatorname{trace}(\mathbf{J}) = u_x + v_y + w_z \tag{2.17}$$

and denotes the gain or loss of mass density at a certain point of the vector field: given a volume element in a flow, a certain amount of mass is entering and exiting it. Divergence is the net flux of this at the limit of a point. A flow field with $\operatorname{div}(\mathbf{v}) = 0$ is called *divergence-free*, which is a common case in fluid dynamics since a number of fluids are *incompressible*.

The *vorticity* or *curl* of a flow field is given as

$$\omega = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \nabla \times \mathbf{v} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix}. \tag{2.18}$$

This vector is the axis of locally strongest rotation, i.e., it is perpendicular to the plane in which the locally highest amount of circulation takes place. The vorticity magnitude $|\omega|$ gives the strength of rotation and is often used to identify regions of high vortical activity. A vector field with $\omega = \mathbf{0}$ is called *irrotational* or *curl-free*, with the important subclass of *conservative* vector fields, i.e., vector fields which are the gradient of a scalar field.

The identification of vortices is a major subject in fluid dynamics. The most widely used quantities for detecting vortices are based on a decomposition of the Jacobian matrix $\mathbf{J} = \mathbf{S} + \Omega$ into its symmetric part, the strain tensor

$$\mathbf{S} = \frac{1}{2}(\mathbf{J} + \mathbf{J}^T) \tag{2.19}$$

and its antisymmetric part, the vorticity tensor

$$\Omega = \frac{1}{2}(\mathbf{J} - \mathbf{J}^T) = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}, \tag{2.20}$$

with ω_i being the components of vorticity (2.18). While Ω assesses vortical activity, the strain tensor $\mathbf{S}$ measures the amount of stretching and folding which drives mixing to occur.

Inherent to the decomposition of the flow field gradient $\mathbf{J}$ into $\mathbf{S}$ and Ω is the following duality: vortical activity is high in regions where Ω dominates $\mathbf{S}$, whereas strain is characterized by $\mathbf{S}$ dominating Ω.

In order to identify vortical activity, Jeong et al. used this decomposition in [JH95] to derive the vortex region quantity λ_2 as the second largest eigenvalue of the symmetric tensor $\mathbf{S}^2 + \Omega^2$. Vortex regions are identified by $\lambda_2 < 0$, whereas $\lambda_2 > 0$ lacks physical interpretation. λ_2 does not capture stretching and folding of fluid particles and hence does not capture the vorticity-strain duality.

The Q-criterion of Hunt [Hun87], also known as the Okubo-Weiss criterion, is defined by

$$Q = \frac{1}{2}(\|\Omega\|^2 - \|\mathbf{S}\|^2) = \|\omega\|^2 - \frac{1}{2}\|\mathbf{S}\|^2. \tag{2.21}$$

Where Q is positive, the vorticity magnitude dominates the rate of strain. Hence it is natural to define vortex regions as regions where $Q > 0$. Unlike λ_2, Q has a physical meaning also where $Q < 0$. Here the rate of strain dominates the vorticity magnitude.

2.2 Parameter-Independent Topology

In this section we collect the first order topological properties of steady 2D and 3D vector fields.

2.2.1 Critical Points

Considering a steady vector field $\mathbf{v}(\mathbf{x})$, an isolated *critical point* $\mathbf{x}_0$ is given by

$$\mathbf{v}(\mathbf{x}_0) = \mathbf{0} \quad \text{with} \quad \mathbf{v}(\mathbf{x}_0 \pm \boldsymbol{\varepsilon}) \neq \mathbf{0}. \tag{2.22}$$

This means that $\mathbf{v}$ is zero at the critical point, but non-zero in a certain neighborhood.

Every critical point can be assigned an *index*. For a 2D vector field it denotes the number of counterclockwise revolutions of the vectors of $\mathbf{v}$ while traveling counterclockwise on a closed curve around the critical point.[1] Similarly, the index of a 3D critical point measures the number of times the vectors of $\mathbf{v}$ cover the area of an enclosing sphere. The index is always an integer and it may be positive or negative. For a curve/sphere enclosing an arbitrary part of a vector field, the index of the enclosed area/volume is the sum of the indices of the enclosed critical points. Mann et al. show in [MR02] how to compute the index of a region using geometric algebra. A detailed discussion of index theory can be found in [FG82, Got90, Got96].

Critical points are characterized and classified by the behavior of the tangent curves around it. In section 2.4 we give a complete classification of critical points based on sectors of different flow behavior around it. Here, we concentrate on first order critical points, i.e., critical points with $\det(\mathbf{J}(\mathbf{x}_0)) \neq 0$. As shown in [HH89, HH91], a first order Taylor expansion of the flow around $\mathbf{x}_0$ suffices to completely classify it. This is done by an eigenvalue/eigenvector analysis of $\mathbf{J}(\mathbf{x}_0)$. Let λ_i be the eigenvalues of $\mathbf{J}(\mathbf{x}_0)$ ordered according to their real parts, i.e., $Re(\lambda_{i-1}) \leq Re(\lambda_i)$. Furthermore, let $\mathbf{e}_i$ be the corresponding eigenvectors, and let $\mathbf{f}_i$ be the corresponding eigenvectors of the transposed Jacobian $(\mathbf{J}(\mathbf{x}_0))^T$.[2] The sign of the real part of an eigenvalue λ_i denotes

[1] For 2D vector fields, it is therefore often called the *winding number*.

[2] Note that $\mathbf{J}$ and $\mathbf{J}^T$ have the same eigenvalues but not necessarily the same eigenvectors.

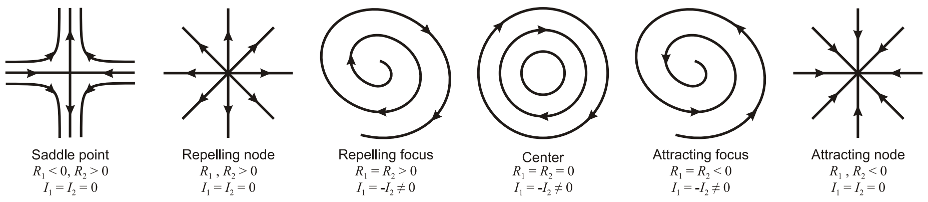


Figure 2.3: Classification of first order critical points. R_1, R_2 denote the real parts of the eigenvalues of the Jacobian matrix while I_1, I_2 denote their imaginary parts (from [HH89]).

– together with the corresponding eigenvector $\mathbf{e}_i$ – the flow direction: positive values represent an *outflow* and negative values an *inflow* behavior. Based on this we give the classification of 2D and 3D first-order critical points in the following.

2D Vector Fields

Based on the flow direction, first order critical points in 2D vector fields are classified into:

$$\begin{array}{lrcccl}
\text{Sources:} & 0 & < & Re(\lambda_1) & \leq & Re(\lambda_2) \\
\text{Saddles:} & Re(\lambda_1) & < & 0 & < & Re(\lambda_2) \\
\text{Sinks:} & Re(\lambda_1) & \leq & Re(\lambda_2) & < & 0
\end{array}$$

Thus, sources and sinks consist of complete outflow/inflow, while saddles have a mixture of both.

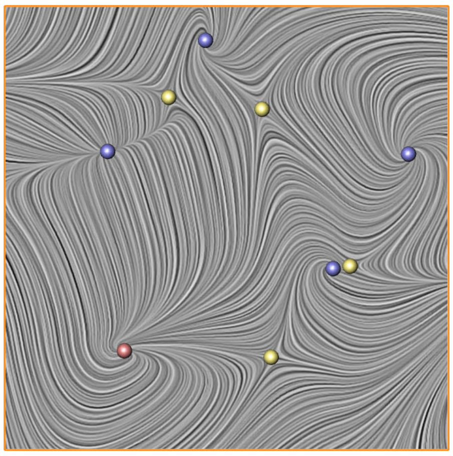
Sources and sinks can be further divided into two stable subclasses by deciding whether or not imaginary parts are present, i.e., whether or not λ_1, λ_2 is a pair of conjugate complex eigenvalues:

$$\begin{array}{ll}
\text{Foci:} & Im(\lambda_1) = -Im(\lambda_2) \neq 0 \\
\text{Nodes:} & Im(\lambda_1) = Im(\lambda_2) = 0
\end{array}$$

There is another important class of critical points in 2D: a *center*. Here, we have a pair of conjugate complex eigenvalues with $Re(\lambda_1) = Re(\lambda_2) = 0$. This type is common in incompressible (divergence-free) flows, but unstable in general vector fields since a small perturbation of $\mathbf{v}$ changes the center to either a sink or a source. Figure 2.3 shows the phase portraits of the different types of first order critical points following [HH89].

The index of a saddle point is -1, while the index of a source, sink, or center is $+1$. It turns out that this coincides with the sign of $\det(\mathbf{J}(\mathbf{x}_0))$: a negative determinant denotes a saddle, a positive determinant a source, sink, or center. This already shows that the index of a critical point cannot be used to distinguish or classify them completely, since different types like sources and sinks have assigned the same index.

An iconic representation is an appropriate visualization for critical points, since vector fields usually contain a finite number of them. Throughout this thesis 2D critical points will be displayed as spheres colored according to their classification: sources will be colored in red, sinks in blue, saddles in yellow, and centers in green (Figure 2.4).


(a) Sources (red), sinks (blue) and saddles (yellow) in an example vector field.

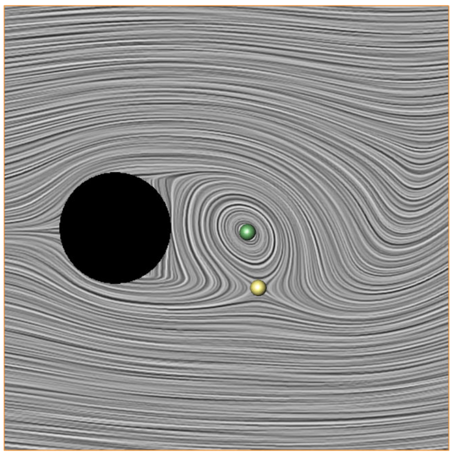

(b) Incompressible flow around a cylinder with a saddle (yellow) and a center (green). Data courtesy of Gerd Mutschke (FZ Rossendorf) and Bernd R. Noack (TU Berlin).

Figure 2.4: Critical points of 2D vector fields.

3D Vector Fields

Depending on the sign of $Re(\lambda_i)$ we get the following classification of first-order critical points in 3D vector fields:

$$\begin{array}{ll}
\text{Sources:} & 0 < Re(\lambda_1) \leq Re(\lambda_2) \leq Re(\lambda_3) \\
\text{Repelling saddles:} & Re(\lambda_1) < 0 < Re(\lambda_2) \leq Re(\lambda_3) \\
\text{Attracting saddles:} & Re(\lambda_1) \leq Re(\lambda_2) < 0 < Re(\lambda_3) \\
\text{Sinks:} & Re(\lambda_1) \leq Re(\lambda_2) \leq Re(\lambda_3) < 0
\end{array}$$

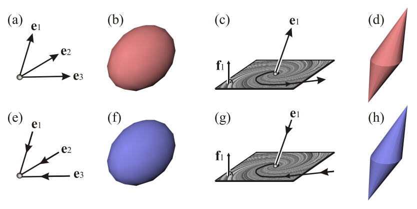
Again, sources and sinks consist of complete outflow/inflow, while saddles have a mixture of both. A repelling saddle has one direction of inflow behavior (called *inflow direction*) and a plane in which a 2D outflow behavior occurs (called *outflow plane*). Similar to this, an attracting saddle consists of an *outflow direction* and an *inflow plane*.

Each of the 4 classes above can be further divided into two stable subclasses by deciding whether or not imaginary parts in two of the eigenvalues are present ($\lambda_1, \lambda_2, \lambda_3$ are not ordered):

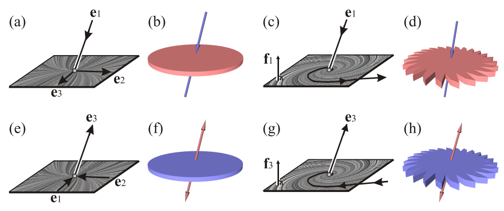
$$\begin{array}{ll}
\text{Foci:} & Im(\lambda_1) = 0 \quad \text{and} \quad Im(\lambda_2) = -Im(\lambda_3) \neq 0 \\
\text{Nodes:} & Im(\lambda_1) = Im(\lambda_2) = Im(\lambda_3) = 0
\end{array}$$

As argued in [GTS04b], the index of a first order critical point is given as the sign of the product of the eigenvalues of $\mathbf{J}(\mathbf{x}_0)$. This yields an index of $+1$ for sources and attracting saddles, and an index of -1 for sinks and repelling saddles.

In order to depict 3D critical points, several icons have been proposed in the literature, see [HH91, GLL91, LDG98, HG00, TWHS03, WTHS04a]. Basically, we follow the design approach of [TWHS03, WTHS04a] and color the icons depending on the flow behavior: Attracting parts (inflow) are colored blue, while repelling parts (outflow) are colored red (Figure 2.5).



(a) Sources and sinks; (a) repelling node and (b) its icon; (c) repelling focus and (d) its icon; (e) attracting node and (f) its icon; (g) attracting focus and (h) its icon.



(b) Repelling and attracting saddles; (a) repelling node saddle and (b) its icon; (c) repelling focus saddle and (d) its icon; (e) attracting node saddle and (f) its icon; (g) attracting focus saddle and (h) its icon.

Figure 2.5: Flow behavior around critical points of 3D vector fields and corresponding iconic representation.

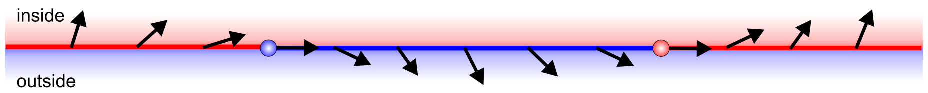


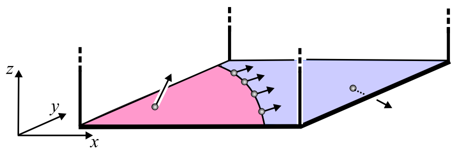
Figure 2.6: Boundary of a 2D vector field **v** consisting of two inflow parts (red) and one outflow part (blue), which are separated from each other by two boundary switch points. The left one is an outbound point (blue) since **v** points into the outflow part at this point. The right boundary switch point is an inbound point (red).

2.2.2 Boundary Switch Points and Curves

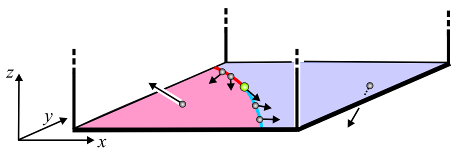
Let **v** be a vector field defined in the domain $D \subset \mathbb{E}^n$ and B be the closed boundary of D. Under the assumption that the flow is allowed to pass through that boundary, one can distinguish between areas of inflow and outflow on B. These areas are separated from each other by *boundary switch points* (2D) or *boundary switch curves* (3D) – being all locations on the boundary where the flow direction is tangential to the boundary. Since these points or curves separate regions of different flow behavior, they are the seeding structures of separatrices emanating from the boundary (see section 2.2.3). In the following we will discuss the properties of boundary switch points/curves for 2D and 3D vector fields separately.

2D Vector Fields: Boundary Switch Points

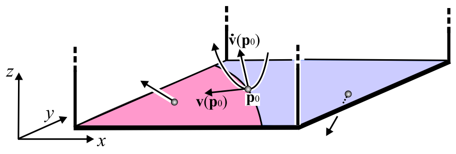
The boundary of a 2D vector field **v** consists of *inflow parts* and *outflow parts*, where **v** points into the domain or out of the domain respectively. These parts are separated from each other by points where **v** is tangential to the boundary, i.e., **v** points neither into the domain nor out of the domain. These points are called *boundary switch points*. Two different types exist depending on the behavior of a stream line through it: either the stream line stays inside or outside the domain D in forward and backward integration. Hence, these points are called inbound and outbound points. Following [dLvL99a], an inbound point is characterized by **v** pointing into the direction of an inflow part. Analogously, at an outbound point, **v** points into the direction of an outflow part. Figure 2.6 gives an illustration.



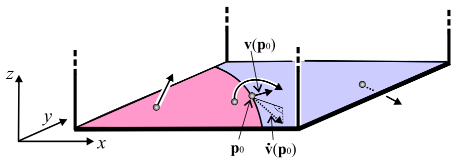
(a) Boundary plane $z = z_{min}$ consisting of an inflow area (red), an outflow area (blue), and their separating boundary switch curve. Shown are 4 vectors of $\mathbf{v}$ on the boundary switch curve, and one each in the inflow and outflow area.



(b) Boundary switch curve consisting of one inbound segment (dark red) and one outbound segment (dark blue). They are separated by an inout point (green).



(c) Inbound point $\mathbf{p}_0$ on a boundary switch curve: $\mathbf{v}(\mathbf{p}_0)$ points into the inflow area, $\dot{\mathbf{v}}(\mathbf{p}_0)$ points inside D. Shown is a part of the stream line starting in $\mathbf{p}_0$ both in forward and backward integration.



(d) Outbound point $\mathbf{p}_0$ on a boundary switch curve: $\mathbf{v}(\mathbf{p}_0)$ points into the outflow area, $\dot{\mathbf{v}}(\mathbf{p}_0)$ points outside D. Shown is a stream line close to $\mathbf{p}_0$ starting in the inflow area and leaving D in the outflow area.

Figure 2.7: Properties of boundary switch curves.

3D Vector Fields: Boundary Switch Curves

In order to ease the explanations in the following, we consider a 3D vector field $\mathbf{v}$ from (2.7) in the domain

$$D = (x_{min}, x_{max}) \times (y_{min}, y_{max}) \times (z_{min}, z_{max}) \tag{2.23}$$

with $x_{min} < x_{max}$, $y_{min} < y_{max}$, $z_{min} < z_{max}$. We assume that no critical point of $\mathbf{v}$ lies on the boundary surfaces of D. Furthermore, D might be the whole domain in which $\mathbf{v}$ is defined, or it may be interactively modified and moved around in the data set, leading to a “local topology” [SHJK00].

The boundary surfaces of D (which are the 6 faces of the bounding box) consist of outflow and inflow areas which are separated by *boundary switch curves*, i.e., all points on the boundary where the flow direction is tangential to the boundary surface. Figure 2.7a illustrates an example of the boundary plane $z = z_{min}$ consisting of one inflow and one outflow area. (In the following we illustrate the concept of boundary switch curves only on the boundary plane $z = z_{min}$. Similar statements hold for the 5 remaining boundary planes of D.)

In general, boundary switch curves do not intersect each other. The case of intersecting boundary switch curves can be considered to be structurally unstable: a small perturbation of $\mathbf{v}$ removes the intersection. Because of this, intersections of boundary switch curves are not considered here.

Given a point $\mathbf{p}_0$ on a boundary switch curve, two cases are possible concerning the stream line starting at $\mathbf{p}_0$:

- Starting from $\mathbf{p}_0$, the stream line integration moves inside D for both backward and forward integration. We call this point an *inbound point* on the boundary switch curve (Figure 2.7c).

- Starting from $\mathbf{p}_0$, the stream line integration moves outside D for both backward and forward integration. Therefore, this stream line in D consists only of $\mathbf{p}_0$ itself. We call this point an *outbound point* (Figure 2.7d).

To distinguish inbound from outbound points, two approaches can be used:

1. Decide whether $\mathbf{v}(\mathbf{p}_0)$ points into the inflow or the outflow area. If $\mathbf{v}(\mathbf{p}_0)$ points into the inflow area, $\mathbf{p}_0$ is an inbound point. If $\mathbf{v}(\mathbf{p}_0)$ points into the outflow area, $\mathbf{p}_0$ is an outbound point. This condition corresponds to the classification of boundary switch points for 2D vector fields given in [dLvL99a].
2. Consider the second derivative vector $\dot{\mathbf{v}}$ of the stream lines as a local property of $\mathbf{v}$ (assuming that the tangent vectors of the stream lines are given by $\mathbf{v}$). This gives
$$\dot{\mathbf{v}}(x,y,z) = \nabla\mathbf{v}\cdot\mathbf{v} = u\,\mathbf{v}_x + v\,\mathbf{v}_y + w\,\mathbf{v}_z. \tag{2.24}$$
See [TF97] and [WT02] for details of this. ($\dot{\mathbf{v}}$ can for example be used to compute the curvature of the stream lines in every point of D: $\kappa = \frac{\|\mathbf{v}\times\dot{\mathbf{v}}\|}{\|\mathbf{v}\|^3}$.) Then $\mathbf{p}_0$ is an inbound point if $\dot{\mathbf{v}}(\mathbf{p}_0)$ points into D. If $\dot{\mathbf{v}}(\mathbf{p}_0)$ points out of D, $\mathbf{p}_0$ is an outbound point.

Figures 2.7c and 2.7d illustrate both criteria. It can be shown by a straightforward exercise in algebra that both criteria 1 and 2 are equivalent. We preferred to use condition 2, since for a domain D given in (2.23), it turns out to be simply a sign check of one component of $\dot{\mathbf{v}}$.

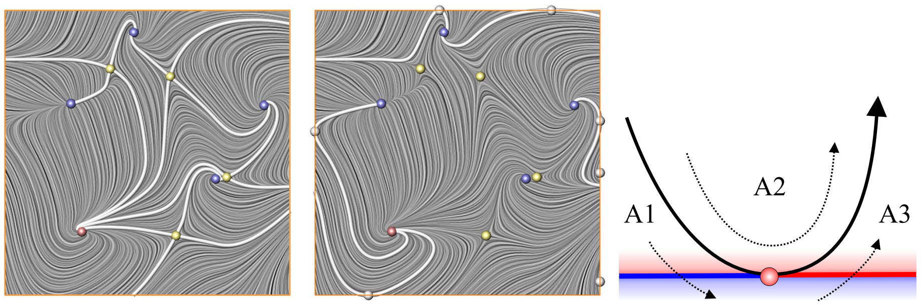
Inbound points and outbound points form *inbound segments* and *outbound segments* on the boundary switch curve. These segments are separated by *inout points*. A point $\mathbf{p}_0$ is an inout point if $\mathbf{v}(\mathbf{p}_0)$ is parallel to the tangent direction of the boundary switch curve in $\mathbf{p}_0$ (or equivalently, if $\dot{\mathbf{v}}(\mathbf{p}_0)$ lies in the tangent plane of the considered boundary of D). One needs to extract all inout points in order to divide the curve into a number of inbound and outbound segments (Figure 2.7b).

The distinction between inbound segments and outbound segments plays an important role for the topological segmentation of a 3D vector field, because the inbound segments are the seeding curves of the separation surfaces emanating from the boundary of D. Outbound segments do not contribute to separation surfaces in the 3D flow.

2.2.3 Separatrices

Separatrices are stream lines or stream surfaces which separate regions of different flow behavior. Different kinds of separatrices are possible: They can emanate from critical points, boundary switch curves, attachment and detachment lines, or they are closed separatrices without a specific emanating structure.

Due to the homogeneous flow behavior around sources and sinks (either a complete outflow or inflow), they do not contribute to separatrices. Each saddle point creates two separatrices: one in forward and one in backward integration into the directions of the eigenvectors. For a 2D saddle point this gives two separation lines (Figure 2.8a). Considering a repelling saddle $\mathbf{x}_R$ of a 3D vector field, it creates one separation curve (which is a stream line starting in $\mathbf{x}_R$ in the inflow direction by backward integration) and a separation surface (which is a stream surface starting in the outflow plane by forward integration). Figure 2.9a gives an illustration. A similar statement holds for attracting saddles.

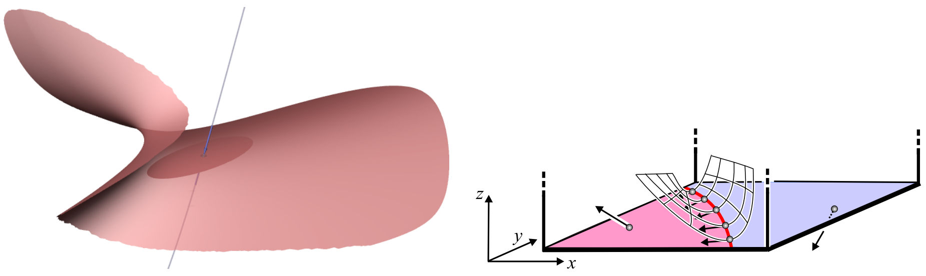


(a) Two separation lines emanate from each saddle into the directions of the eigenvectors: one integrated in forward and the other one in backward integration.

(b) Separation lines starting from boundary switch points (gray). Only inbound points contribute to separation.

(c) Separation lines starting from the boundary divide the domain into three areas A_1, A_2, A_3.

Figure 2.8: Separatrices of a 2D vector field starting from saddles and boundary switch points.



(a) Separatrices of a 3D vector field originating from a repelling node saddle.

(b) Separation surface originating from an inbound segment of a boundary switch curve.

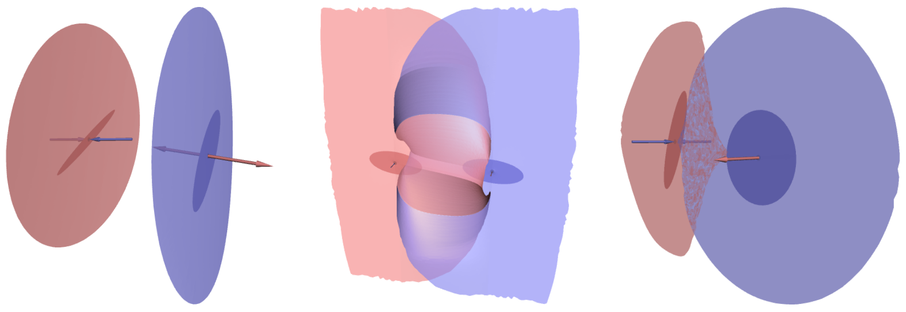
Figure 2.9: Separatrices of a 3D vector field starting from saddles and boundary switch curves.

Considering a stream line starting from a boundary switch point or curve, the stream line integration moves either inside or outside of the domain. Hence, only inbound points or segments play a role for the segmentation of regions of different flow behavior inside the domain (see section 2.2.2). Figure 2.8b shows separation lines starting from boundary switch points in a 2D vector field. Figure 2.9b illustrates a separation surface starting from an inbound segment in a 3D vector field. Note that inbound points or segments always create two separatrices: one in forward and one in backward integration. They divide the domain into three areas as shown in Figure 2.8c.

Another type of separatrices, namely closed stream lines, will be discussed in section 2.2.6.

2.2.4 Saddle Connectors

This section introduces the new concept of saddle connectors, which has been developed in the course of this thesis [TWHS03]. In contrast to 2D vector fields, the set of separatrices of 3D vector fields consists also of stream surfaces (Section 2.2.3) – a fact



(a) No intersection. (b) Two intersection curves. (c) The separation surfaces collapse.

Figure 2.10: Intersection of separation surfaces.

which creates a number of problems. In particular, we see the following challenges:

- The integration of stream surfaces is computationally more involved and less stable than the integration of stream lines, since convergence and divergence effects on the stream surface may occur.

- The visualization of the topological skeleton of a vector field requires simultaneous visualization of a higher number of stream surfaces, which very soon leads to visually cluttered representations, since the surfaces hide each other as well as the critical points. This problem remains, even if the separation surfaces are rendered in a semi-transparent mode.

A number of solutions have been proposed for the first problem, see [Hul92, Gel01, SBM$^+$01, vW93].

In this work, we tackle the second problem. In order to create sparse visual representations that avoid occlusion and minimize visual clutter, we propose to represent the separation surfaces as a finite number of stream lines. These stream lines are the intersection curves of the separation surfaces. We call them *saddle connectors* because they start and end in saddle points of the vector field.

The saddle connectors indicate only the approximate run of the separation surfaces and of course cannot completely substitute them. A possible procedure is to always depict the saddle connectors and interactively, on user demand, additionally display *single* separating surfaces.

The basic idea of saddle connectors is to consider the intersection of the separation surfaces of two saddle points. For this intersection, the following cases are possible:

- The separation surfaces of two saddles have no intersection (Figure 2.10a).

- The separation surfaces have one intersection curve (Figure 2.11a).

- The separation surfaces have more than one, but a finite number of intersection curves (Figure 2.10b).

- The separation surfaces partially collapse. In this case, the intersection of the separation surfaces is a surface (Figure 2.10c).

We define saddle connectors as follows (Figure 2.11 gives an illustration):

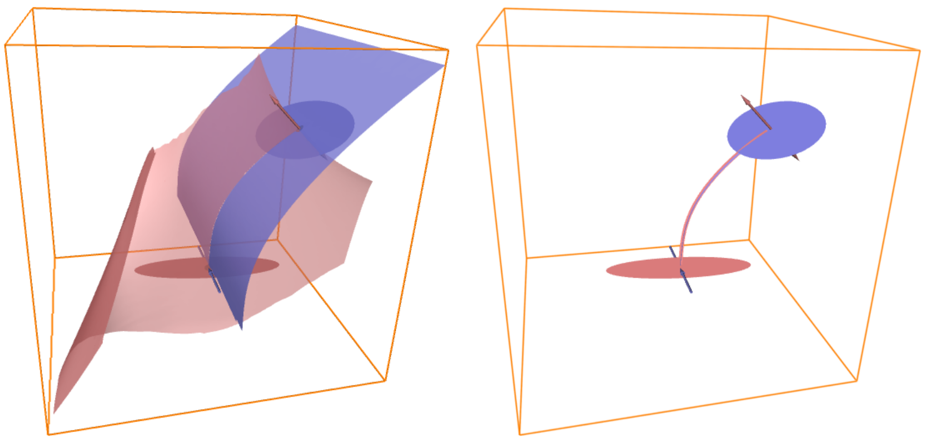

(a) Separation surfaces of the saddles.
(b) The intersection of the separation surfaces is the saddle connector.

Figure 2.11: Definition of saddle connectors.

Definition 1 *Let* $\mathbf{v}$ *be a 3D vector field, and let* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *be two saddle points in* $\mathbf{v}$. *We consider the intersection of the two separation surfaces starting in the outflow/inflow planes of* $\mathbf{x}_1$ *and* $\mathbf{x}_2$. *If this intersection is a curve, we call it a saddle connector.*

Note that this definition excludes cases of partially collapsing separation surfaces. This is justified by the fact that this case can be seen as an unstable situation in the vector field.

An intersection of the separating surfaces of two saddle points can only exist if one of the saddles is an attracting saddle and the other one is a repelling saddle. To see this, imagine for instance two attracting saddles[3] $\mathbf{x}_{A1}$, $\mathbf{x}_{A2}$, and suppose that a certain point $\mathbf{p}$ lies on both separation surfaces of $\mathbf{x}_{A1}$ and $\mathbf{x}_{A2}$. Then the stream line starting from $\mathbf{p}$ in forward direction must both pass through $\mathbf{x}_{A1}$ and $\mathbf{x}_{A2}$, which contradicts to basic properties of critical points and stream lines.

Also from definition 1 we obtain that a saddle connector is a stream line which starts in the outflow plane of a repelling saddle $\mathbf{x}_R$ and ends in the inflow plane of an attracting saddle $\mathbf{x}_A$. This holds because for every stream surface, the stream line starting from any point on this surface lies completely in the stream surface. Thus, if a point $\mathbf{p}$ lies on both separation surfaces of $\mathbf{x}_R$ and $\mathbf{x}_A$, the whole stream line starting in $\mathbf{p}$ in forward and backward direction lies in both separation surfaces. Therefore, this stream line connects both saddles.

One way of analyzing separation surfaces is asking for their boundary curves. If a separation surface does not have a strong diverging behavior, its boundary curve gives a good deal of information about its behavior in the 3D domain of $\mathbf{v}$. The boundary curve of a separation surface may be a closed curve on the boundary of the domain of $\mathbf{v}$. It is also possible that the separation surface ends in a number of sinks *or* sources. In this case, there is a relation between the saddle connectors and the boundary curves of the separation surfaces. To compute the boundary curve of the separation surface of a repelling saddle $\mathbf{x}_R$, we can compute the saddle connectors of $\mathbf{x}_R$ with all attracting saddles. Then we consider the repelling separation curves of all attracting saddles which share a saddle connector with $\mathbf{x}_R$. If the union of all these curves forms a closed

[3]Or two repelling saddles.

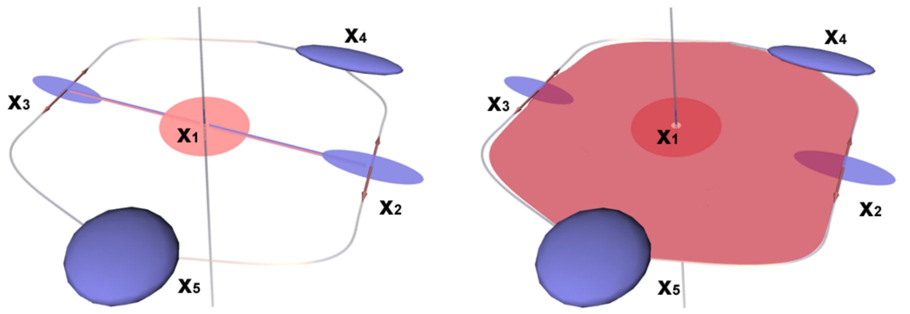


Figure 2.12: (left) The repelling saddle $\mathbf{x}_1$ has saddle connectors to the attracting saddles $\mathbf{x}_2$ and $\mathbf{x}_3$. The repelling separation curves of $\mathbf{x}_2$ and $\mathbf{x}_3$ end in the sinks $\mathbf{x}_4$ and $\mathbf{x}_5$, and form a closed curve. (right) The separation curves of $\mathbf{x}_2$ and $\mathbf{x}_3$ are the boundary curves of the separation surface of $\mathbf{x}_1$.

curve, this closed curve describes the boundary curve of the separation surface of $\mathbf{x}_R$. Figure 2.12 shows an example.

2.2.5 Boundary Switch Connectors

As already discussed in section 2.2.3, each inbound segment of a boundary switch curve creates two separation surfaces: one is obtained by applying a forward integration starting from the boundary switch curve, the other one by backward integration. The visualization of these separation surfaces creates the same problems as identified in section 2.2.4 for separation surfaces starting from saddles: if a higher number of separation surfaces is present, their visualization tends to be cluttered due to various occlusion effects.

A solution for this problem is the extension of the idea of saddle connectors: instead of visualizing the separation surfaces directly, we compute all intersection curves of these surfaces and visualize this skeleton of curves. In fact, we choose a general approach which yields the intersection curves of all separation surfaces starting either from a saddle point or a boundary switch curve. Analyzing these curves, we obtain the following properties:

- The intersection curves of two separation surfaces are stream lines. This is due to the fact that the intersection of two stream surface is always a stream line (or degenerate).
- Each intersection curve starts either in the outflow plane of a repelling saddle, or on a boundary switch curve by integrating in forward direction.
- Each intersection curve ends either in the inflow plane of an attracting saddle or on a boundary switch curve by integrating in backward direction.

The latter two statements give the following classification regarding the intersection curves of the separation surfaces:

1. The curve starts in a repelling saddle and ends in an attracting saddle. (Figure 2.11)

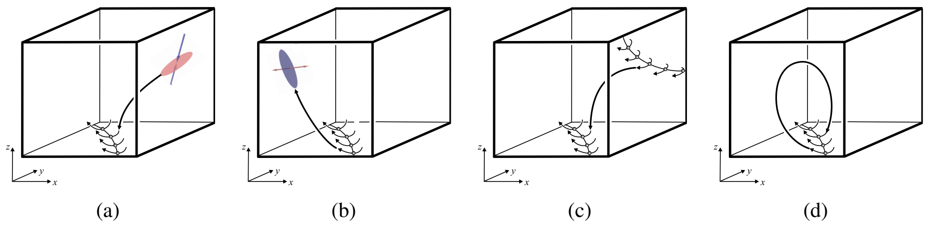
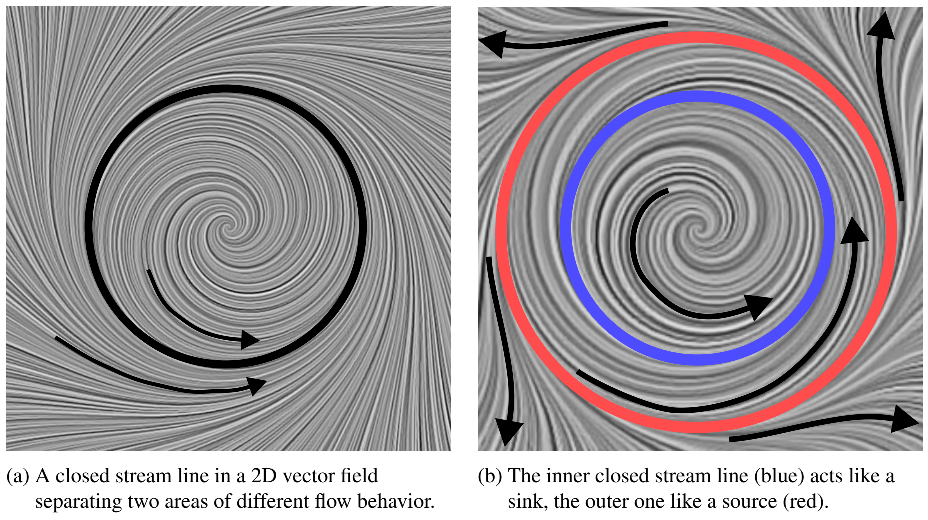


Figure 2.13: Boundary switch connectors are the intersection curves of separation surfaces where at least one surface starts from the boundary.



(a) A closed stream line in a 2D vector field separating two areas of different flow behavior.

(b) The inner closed stream line (blue) acts like a sink, the outer one like a source (red).

Figure 2.14: Closed stream lines of 2D vector fields.

2. The curve starts in a repelling saddle and ends in a boundary switch curve. (Figure 2.13a)
3. The curve starts in a boundary switch curve and ends in an attracting saddle. (Figure 2.13b)
4. The curve starts in a boundary switch curve and ends in a different boundary switch curve. (Figure 2.13c)
5. The curve starts in a boundary switch curve and ends in the same boundary switch curve. (Figure 2.13d)

Case 1 refers to saddle connectors as treated in section 2.2.4. Cases 2–5 refer to curves starting or ending in boundary switch curves. Therefore, we call them *boundary switch connectors*. Figure 2.13 illustrates all possible types of boundary switch connectors.

2.2.6 Closed Stream Lines

Isolated closed stream lines (or periodic orbits) of a 2D vector field are important global topological structures since they separate the vector field into two areas of different

Feature	$\mathbf{v}(\mathbf{x})$	$\mathbf{v}(\mathbf{x},t)$	$\mathbf{v}(\mathbf{x},s,t)$
critical points	points	curves	surfaces
fold / Hopf bifurcations	*n/a*	points	curves
fold-fold / Hopf-fold bifurcations	*n/a*	*n/a*	points

Table 2.1: Dimensionality of topological features for vector fields depending on zero, one or two parameters. Entries marked with "*n/a*" are either not available in this dimension or structurally unstable.

flow behavior: inside and outside the closed stream line (Figure 2.14a). Furthermore, they are indicators of recirculating flow behavior. Their influence on the flow is either (Figure 2.14b):

- **sink-like**, i.e., all stream lines close to the closed stream line *converge* to it without actually reaching it.
- **source-like**, i.e., all stream lines close to the closed stream line *diverge* from it.

Closed stream lines can be found in 3D vector fields as well, but are not treated in this thesis. An analysis of these structures can be found in [Asi93, PS07].

2.3 Parameter-Dependent Topology

The topology of parameter-dependent vector fields $\mathbf{v}(\mathbf{x},\mathbf{a})$ builds upon the topological properties of steady vector fields $\mathbf{v}(\mathbf{x})$, i.e., for a fixed set of parameters $\mathbf{a} = \text{const.}$ the topological structures are identical to the corresponding steady vector field $\mathbf{v}(\mathbf{x})$ as discussed in section 2.2. However, with a changing set of parameters $\mathbf{a}$ the topological structures will change their positions and properties as well. As an example, consider critical points in a vector field $\mathbf{v}(\mathbf{x},t), \mathbf{x} \in \mathbb{R}^n$ depending on one parameter t (like e.g. a time-dependent flow): critical points move with changing t forming line structures in $\mathbb{R}^{n+1}$. In a two-parameter-dependent vector field they form surface structures. See Table 2.3 for an overview of the dimensionality of topological features.

Studying the topology of parameter-dependent vector fields means to find the occurrences of all topological features for the complete parameter space $\mathbf{a} \in \mathbb{E}^p$. In order to understand the dynamics of the data, the following has to be taken into account:

- **Correspondence**:
  One has to find the correspondence between features of different parameter settings.
- **Bifurcations**:
  Features might also abruptly change their properties, or they might abruptly appear or disappear at some parameter setting. Such structural changes are called *bifurcations* or *events*.

The correspondence problem can be solved in a number of ways. This will be discussed in chapter 5. In the following two sections we will discuss the types of bifurcations that may occur in parameter-dependent vector fields. We focus on structurally stable bifurcations needed to understand the dynamics of flow fields. A detailed introduction to the theory of bifurcations can be found in [GH86].

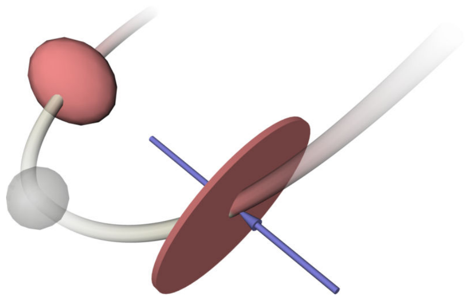


Figure 2.15: A repelling saddle and a source in a 3D time-dependent vector field shortly before they merge and disappear in a fold bifurcation (gray, semi-transparent sphere). The paths of the moving critical points are shown as semi-transparent lines encoding past and future.

Note that the main motivation behind topological methods is to segment a vector field into areas of similar flow behavior which is determined by observing the behavior of certain characteristic curves. For a subset of one-parameter-dependent vector fields, namely time-dependent vector fields, two classes of locally unique curves exist (see section 2.1.2): stream lines and path lines. Hence, for time-dependent vector fields we can distinguish between two different kinds of topologies: a stream line oriented topology where areas are segmented which show a similar behavior of stream lines, and a path line oriented topology which does so for path lines. In this work we will mainly discuss the stream line oriented topology where time is considered as a parameter of the dynamical system. In section 8.1 we will briefly review path line oriented approaches.

The actual visualization of parameter-dependent fields is another challenge since it usually requires to depict features in spaces with more than three dimensions. We will discuss different approaches in the course of the following sections.

2.3.1 Bifurcations of One-Parameter-Dependent Vector Fields

An example of an one-parameter-dependent vector field is a time-dependent vector field $\mathbf{v}(\mathbf{x},t), \mathbf{x} \in \mathbb{R}^n$. In order to ease the explanation, we use this example in the following. Furthermore, the stream lines of this field are described by $\mathbf{s} = (\mathbf{v}, 0)^T$ as given by (2.14).

As already mentioned, critical points move over time in time-dependent vector fields – thereby forming line structures. Furthermore, critical points might appear or disappear at some t. Those births and deaths of critical points are called *fold bifurcations* and their occurrence changes the topological behavior of the field abruptly: two critical points of opposite index collapse and disappear (or the other way around: a critical point appears and splits up into two critical points immediately). Figure 2.15 illustrates this.

Critical points might also enter or leave the domain. These events are called *entry* and *exit* respectively.

Another important structural change is a so-called *Hopf bifurcation*. It occurs at locations where two of the eigenvalues of the Jacobian matrix $\mathbf{J}(\mathbf{v})$ are purely imaginary, i.e., their real parts are vanishing. In a 2D time-dependent vector field this denotes a spiraling source changing to a spiraling sink or vice versa. In a 3D time-dependent vector field we have a similar behavior, but it occurs in an appropriate plane through the critical point, where 2D circulating flow is present. Figure 2.16 gives an illustration. Considering this plane only, we can still speak of a spiraling source becoming a spiraling sink or vice versa. Together with the third eigenvalue, this defines the types of critical points that can be part of a Hopf bifurcation in a 3D time-dependent vector field: spiraling source and sink as well as attracting and repelling focus saddle. Since

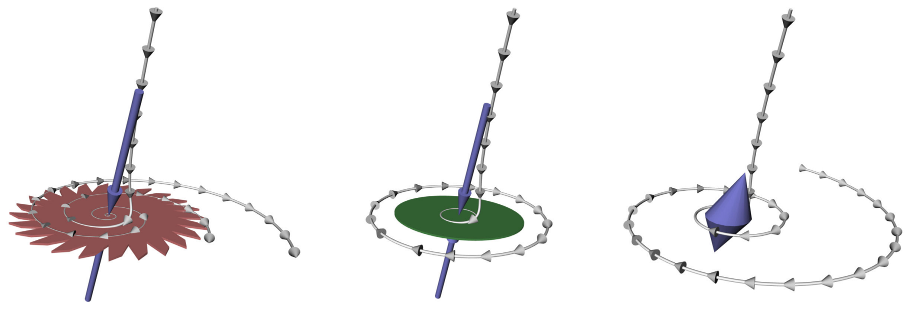

(a) Shortly before. (b) The event. (c) Shortly after.

Figure 2.16: Hopf bifurcation in a 3D vector field where a repelling focus saddle changes to a spiraling sink. The bifurcation occurs in a plane through the critical point, where 2D circulating flow is present.

the index of the critical point has to be preserved, only the following transformations are possible:

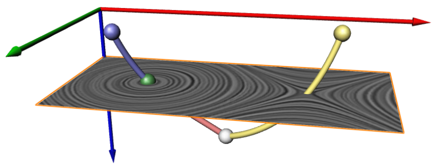
- spiraling source (repelling focus) $\Longleftrightarrow$ attracting focus saddle
- spiraling sink (attracting focus) $\Longleftrightarrow$ repelling focus saddle

Hopf bifurcations trigger the birth or death of closed stream lines.

Figure 2.17a visualizes the topological skeleton of a 2D time-dependent vector field. Here we use a space-time diagram to encode the temporal evolution: time is explicitly shown as the third dimension and denoted by the blue axis. For the topological features we use the same color coding as introduced in section 2.2.1: the critical lines of **s** are color coded according to the inflow/outflow behavior of the represented critical points in **v**: a red/blue/green/yellow line segment represents a source/sink/center/saddle critical point respectively. The same color coding is used for particular critical points which are visualized as small spheres. This means that a Hopf bifurcation is shown as a small green sphere. Furthermore, fold bifurcations are shown as gray spheres, while particular stream lines of **s** are shown as gray lines. Figure 2.17b shows another part of the topological skeleton of **v**: the separation curves starting from saddle points. It is a well-known fact that a saddle of a 2D vector field creates four separation curves by starting the integration into the directions of the eigenvectors of the Jacobian matrix. While the saddle moves over time in **v**, their sweepings form four stream surfaces dividing **s** into areas of different flow behavior. We color code these separation surfaces according to the integration direction as red (forward integration) or blue (backward integration).

The moving critical points of a 3D time-dependent vector field form 4D line structures. In order to visualize them, we allow the user to change the current time step interactively. For this given t, the structures are displayed as in the 3D steady case. Additionally, we display the past and future of the critical points as faded lines. This is shown in figure 2.15 where two critical points are depicted shortly before they disappear in a fold bifurcation. However, in complex settings the faded lines may become rather cluttered and can be switched off.

The above discussed bifurcations are local events. They allow to detect all important structural changes related to critical points. The treatment of 3D vector fields will


(a) Critical lines of **s**, LIC plane through Hopf bifurcation.

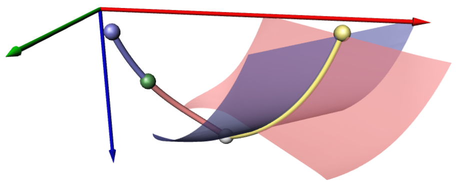

(b) Separation surfaces created by the moving saddle.

Figure 2.17: Topological visualization of a simple 2D time-dependent vector field consisting of sink, source, saddle, fold and Hopf bifurcation – one of each type.

be limited to these kinds of bifurcations. However, for 2D time-dependent vector fields we also discuss the following global bifurcations.

Saddle connections are global bifurcations which appear when two separatrices starting from saddle points coincide, i.e., when a separatrix of one saddle ends in another saddle. Figure 2.18 illustrates an example. A special case of saddle connections is the so-called *periodic blue sky bifurcation* ([AS92]) where two separatrices of the same saddle coincide. Note that this is a special case of a closed stream line, too. Figure 2.19 illustrates this.

Closed stream lines are global topological features which evolve over time in **v** (see section 2.2.6). Several bifurcations can occur: a closed stream line may enter or exit the domain, or two closed stream lines may collapse and disappear. The latter case is called *cyclic fold bifurcation* and is illustrated in Figure 2.20.

Figure 2.21 shows all three types of global bifurcations in the corresponding 3D space-time diagram. The green surface in figure 2.21c denotes the evolution of two closed stream lines coming closer to each other until they merge and disappear in a cyclic fold bifurcation (gray curve). The intersection of the LIC plane with the surface gives the current positions of the closed stream lines. Shown is the same time step as in figure 2.20b.

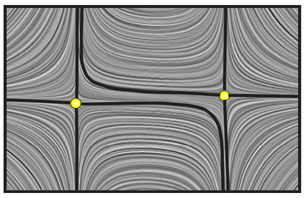
2.3.2 Two-Parameter-Dependent Topology

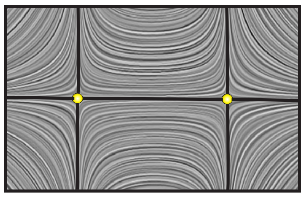
Two-parameter-dependent vector fields are of the form $\mathbf{v}(\mathbf{x},s,t)$ and their stream lines are described by $\mathbf{s} = (\mathbf{v},0,0)^T$ as given by (2.15). Critical points move with changing s and t and form surface structures now. Consequently, bifurcation points move as well and form line structures in $\mathbb{R}^{n+2}$. This creates new types of structurally stable bifurcations: births and deaths of fold or Hopf bifurcations themselves. We call them *fold-fold* and *Hopf-fold* bifurcations. Those events are isolated points in $\mathbb{R}^{n+2}$.

In this thesis we restrict ourselves to extracting all occurrences and properties of all critical points of two-parameter-dependent 2D vector fields, i.e., vector fields of the form $\mathbf{v}(x,y,s,t)$. To do so, only fold-fold and Hopf-fold bifurcations need to be considered. A more in-depth discussion of the these bifurcations is developed alongside the extraction scheme in section 5.3.

An expressive visual representation of the 4D surface structures formed by the moving critical points is a challenging task. We use three approaches:

- Interactively changing s and t: a point (s,t) is interactively moved in the domain $[s_{min}, s_{max}] \times [t_{min}, t_{max}]$, and the topological skeleton of **v** at (s,t) is visualized. Figure 2.22a illustrates this. We refer to this as (x,y)-visualization.


(a) Shortly before.


(b) The event.

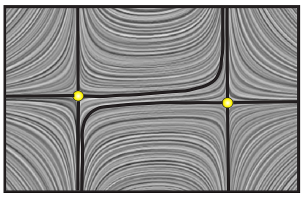
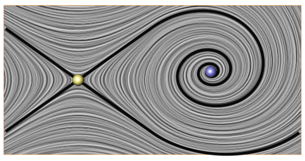
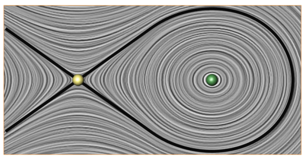

(c) Shortly after.

Figure 2.18: Saddle connection bifurcation.


(a) Shortly before.


(b) The event.

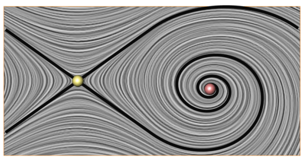

(c) Shortly after.

Figure 2.19: Periodic blue sky bifurcation.

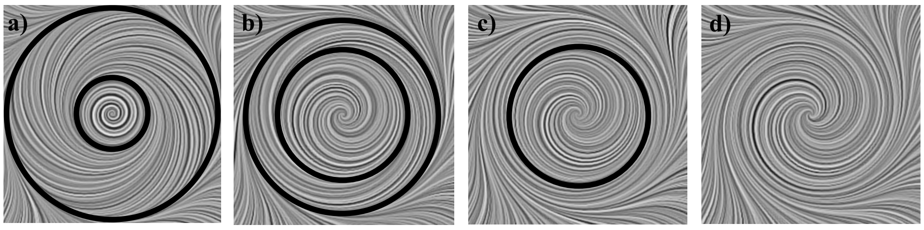
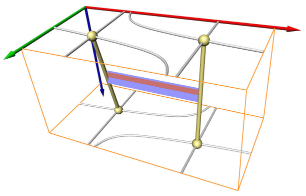
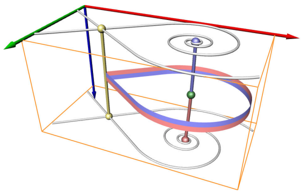


Figure 2.20: Cyclic fold bifurcation: two closed stream lines move towards each other (a and b), merge (c) and disappear (d).


(a) Saddle connection denoted by red-blue band.


(b) Periodic blue sky bifurcation denoted by red-blue band.

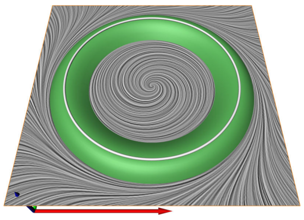
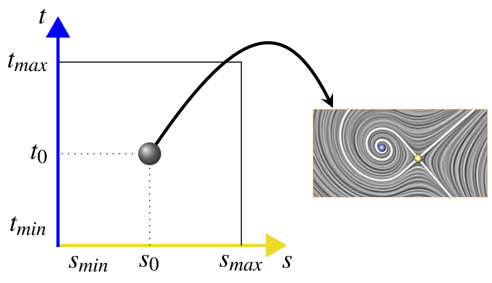
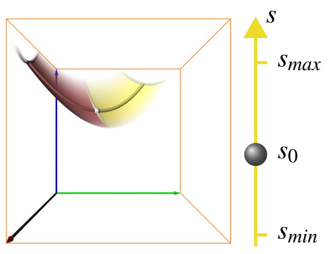

(c) Cyclic fold bifurcation denoted by the gray curve. Compare with 2.20b.

Figure 2.21: Global bifurcations of one-parameter-dependent 2D vector fields visualized in the 3D space-time diagram.



(a) (x,y)-visualization for a fixed (s_0,t_0).



(b) (x,y,t)-visualization for a fixed s_0.

Figure 2.22: Visualization approaches for two-parameter-dependent 2D vector fields.

- Interactively changing s: for a given location s, $\mathbf{v}$ is interpreted and visualized as one-parameter-dependent field $\mathbf{v}(\mathbf{x},t)$. In addition, faded-out surfaces starting from the zero-lines denote the situation in the direct s-past/future. Figure 2.22b gives an illustration. There, the topological skeleton for a fixed s consists of a moving saddle (yellow curve), a moving source (red curve), and a fold bifurcation (gray point). When moving forward/backward in s, these features change shape and location. This is represented by the faded-out red/yellow surfaces for the moving source/saddle, and by the faded-out gray line starting from the fold bifurcation. To avoid cluttered visualizations, the faded lines/surfaces can be switched off. We call this a (x,y,t)-visualization.

- Interactively changing t: similar to the previous approach, but with a one-parameter-dependent visualization of $\mathbf{v}(\mathbf{x},s)$ for a fixed t. We call this a (x,y,s)-visualization.

Note that although the two latter visualization approaches give more insight into the topological behavior of $\mathbf{v}$, the first one is the only one which straightforwardly extends to 3D vector fields depending on two parameters. In all figures, we use the following colors for the coordinate axes: red$=x$, green$=y$, yellow$=s$, blue$=t$.

2.4 Higher Order Topology

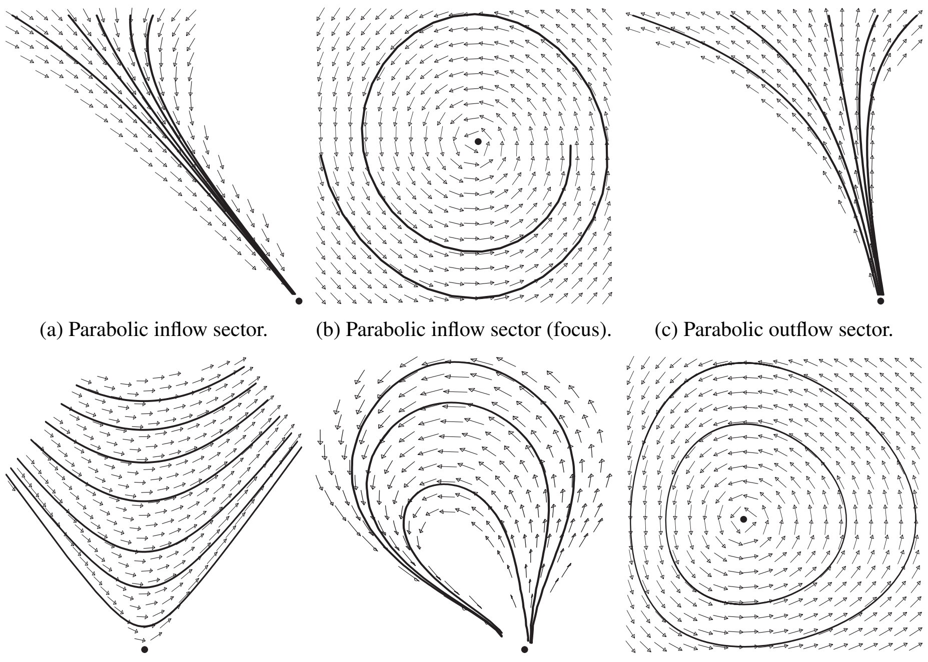
In section 2.2.1 we discussed the topology of first order critical points, i.e., the Jacobian at a critical point $\mathbf{c}$ fulfills $\det(\mathbf{J}(\mathbf{c})) \neq 0$. Under this assumption, $\mathbf{c}$ can be classified as a source, sink or saddle by an eigenvalue/eigenvector analysis of $\mathbf{J}$ as described.

Here we are particularly interested in higher order critical points, i.e., critical points with $\det(\mathbf{J}(\mathbf{c})) = 0$. To capture the topology of such a point $\mathbf{c}$, we have to examine the flow behavior in its neighborhood.[4] Every point $\mathbf{x}$ in this neighborhood can be classified by considering the stream line starting in $\mathbf{x}$ in both forward and backward direction. In each of these directions the stream line can have two kinds of behavior concerning $\mathbf{c}$: it may run into $\mathbf{c}$, or it may diverge away from $\mathbf{c}$. Then $\mathbf{x}$ can be classified in the following way:

[4]In order to capture the topology of $\mathbf{c}$ only, the neighborhood has to be chosen that small that the flow at any point in this neighborhood is governed by $\mathbf{c}$. In particular, no other critical point except $\mathbf{c}$ must be inside this neighborhood.

forward integration	backward integration	F-classification	color scheme	
ends in **c**	**diverges** from **c**	inflow	blue	🟦
diverges from **c**	**ends** in **c**	outflow	red	🟥
diverges from **c**	**diverges** from **c**	hyperbolic	yellow	🟨
ends in **c**	**ends** in **c**	elliptic	green	🟩

Table 2.2: F-Classification: flow behavior of a stream line starting at a point **x** with respect to a nearby critical point **c**.


(a) Parabolic inflow sector. (b) Parabolic inflow sector (focus). (c) Parabolic outflow sector.

(d) Hyperbolic sector. (e) Elliptic sector. (f) Center.

Figure 2.23: Sectors with different flow behavior around a 2D critical point.

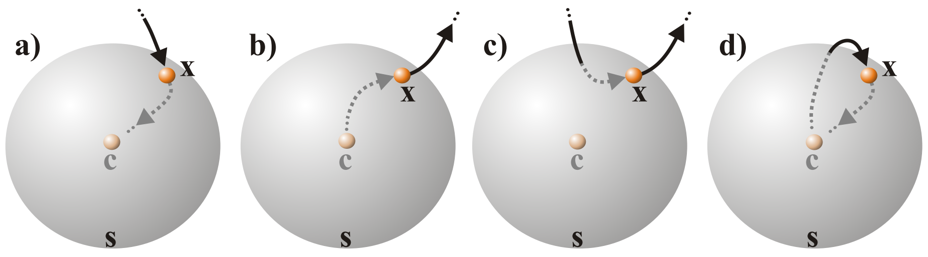


Figure 2.24: The F-classification of a point **x** on a surface **s** around a 3D critical point **c** depends on the behavior of the stream line through **x** in forward and backward direction: **x** belongs to a) an inflow sector, b) an outflow sector, c) a hyperbolic sector, d) an elliptic sector.

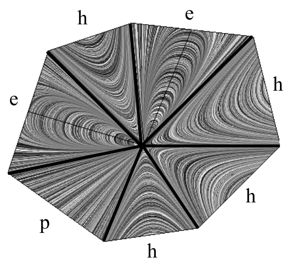


Figure 2.25: Example of a 2D higher order critical point consisting of 4 hyperbolic, 2 elliptic and one parabolic sector (from [The02]).

- **x** belongs to an *inflow sector* (parabolic) if the forward integration of **v** from **x** ends in **c** while the backward integration diverges away from **c**.
- **x** belongs to an *outflow sector* (parabolic) if the forward integration diverges away from **c** and the backward integration ends in **c**.
- **x** belongs to a *hyperbolic sector* if both forward and backward integration diverge away from **c**.
- **x** belongs to an *elliptic sector* if both forward and backward integration end in **c**.

We call this classification the *F-classification* of **x**. “F” stands for flow because we analyze where the flow (both in forward and backward direction) starting from a point converges to. Figures 2.23 and 2.24 illustrate this for 2D and 3D respectively, whereas table 2.2 summarizes it. This table already introduces the color coding which later will be used to visualize areas of different F-classification.

Only a single case does not fit into this classification, namely a center as depicted in 2.23f. This is the only case where a stream line started at **x** neither ends nor diverges from **c**. This case shall be neglected in the further discussions of higher order topological structures.

Describing the topology of a critical point **c** means to find a segmentation of its neighborhood into sectors of different F-classification: a critical point is completely classified by specifying type and position of all sectors around it. This will be described for 2D and 3D vector fields independently in the following sections.

2.4.1 2D Vector Fields

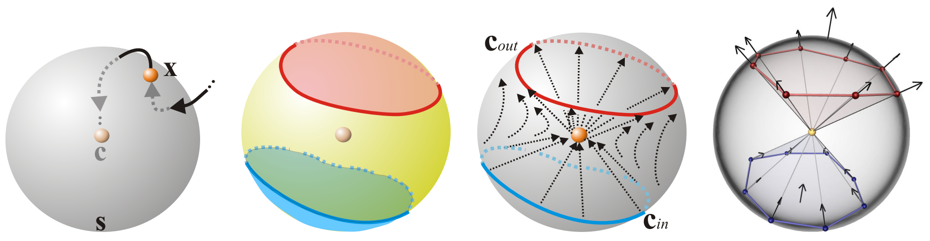
The topology of higher order 2D critical points is well studied [FG82, SHK+97, TSH00, The02, LVRL06]. Different topological sectors are arranged around the critical point and separated from each other by certain tangent curves (separation curves) as exemplified in figure 2.25. A complete classification of a 2D critical point is given by the sequence of topological sectors around it. For example, the higher order critical point in figure 2.25 is classified by (h, e, h, e, p, h, h).

In order to detect regions of different flow behavior around a 2D critical point, it suffices to consider the flow behavior on an enclosing closed polygon [TSH00, The02]. The different topological sectors leave their footprint on this polygon: the sectors become line segments or isolated points, the separation curves become points on the polygon.

The index of a 2D critical point with arbitrary topology is given by

$$\text{index} = 1 + \frac{n_e - n_h}{2} \tag{2.25}$$

where n_e, n_h are the numbers of elliptic and hyperbolic sectors respectively [FG82].



(a) Although $\mathbf{v}(\mathbf{x})$ points outside **s**, **x** is part of an inflow sector.

(b) Three volumetric sectors leaving their footprint on **s**.

(c) Two separation curves separating three sectors.

(d) Constructing the separation surfaces of the icon.

Figure 2.26: F-classification and topological segmentation on a sphere around a 3D critical point.

2.4.2 3D Vector Fields

Profound mathematical studies of the behavior of higher order critical points in 3D exist [Tak74, BD86, DI98], but have not been used for visualization purposes so far. In the course of this thesis, we introduced the following classification of 3D higher order critical points to the Computer Graphics and Visualization community. Furthermore, a corresponding visualization technique and a number of applied methods – simplification (chapter 6) and construction (section 7.1) – have been developed for the first time.

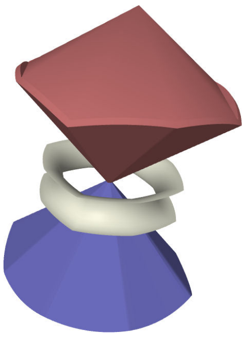
In order to detect all sectors of different flow behavior around a 3D critical point **c**, it suffices to consider a small closed surface **s** around **c** which has a well-defined surface normal almost everywhere:[5] as an example, a volumetric inflow region of **c** leaves its footprint on the surface as an area where all points have an inflow classification. In more general words, an area on **s** where all points have the same F-classification corresponds to a volumetric topological sector around **c** (Figure 2.26b).

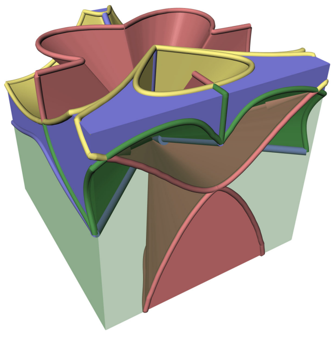
These areas are separated from each other by certain separation curves on **s**. These curves have the property that all points on them have the same F-classification, while the adjacent points on at least one side of the curve have another F-classification. Note that separation curves on **s** can have any type of F-classification, and they do not have to be closed. Examples of separation curves are shown in figure 2.26c. Here the red and blue closed lines on **s** separate a hyperbolic sector from an inflow sector and an outflow sector respectively. Note that the separation curves on **s** can be considered as the footprints of surfaces separating different volumetric sectors. Furthermore, a separation curve on **s** might degenerate to a point – denoting a single stream line with a different F-classification than the surrounding region.

Note that for a classification of a point $\mathbf{x} \in \mathbf{s}$ it is not sufficient to check whether $\mathbf{v}(\mathbf{x})$ points inside or outside **s**. Figure 2.26a shows an example where $\mathbf{v}(\mathbf{x})$ points outside **s** while the stream line integration starting from **x** yields that **x** is part of an inflow sector.

Given this classification of critical points, the well-known first-order critical points fit into this system as well. Sources and sinks consist of one parabolic sector covering the whole surface of **s**. A saddle consists of an outflow plane and two degenerate inflow surfaces (or the other way around, an inflow plane and two degenerate outflow

[5] A variety of different shapes for **s** can be chosen. For the theoretical explanation in this section we use a sphere, whereas later for the extraction in chapter 6 we use a box. In section 7.1 we use the sphere model again in order to model critical points of arbitrary topology.


(a) Simple 3D higher order critical point (**s** is a sphere). Compare with figure 2.26.


(b) Complex 3D higher order critical point (**s** is a box).

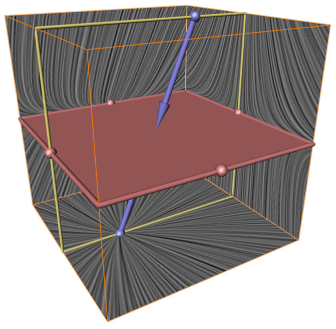

(c) First order saddle point visualized using the scheme for higher order critical points (**s** is a box). Compare with figure 2.5b.

Figure 2.27: Icons of 3D critical points with arbitrary topology.

surfaces). Hence, a saddle consists of two hyperbolic sectors.

To the best of our knowledge, a relation similar to (2.25) between the index of a 3D higher order critical point and the number of topological sectors is still an open issue for future research. However, the index can be numerically computed using a closed surface around the region of interest as shown in [MR02].

To visualize a 3D higher order critical point, we place an appropriate icon into its location. The separation surfaces between the different sectors are constructed by creating a triangle fan between the separation curves on **s** and the critical point **c** (Figure 2.26d. If a separation surface degenerates to a line, i.e., a point on **s**, we visualize it using a 3D arrow. Color coding is according to the F-Classification. It remains to visually depict the sectors of different 3D flow behavior. Inflow/outflow sectors are represented by filling the areas with an opaque surface using the respective color. For elliptic and hyperbolic sectors we have two alternative schemes. For rather simple topologies we display a white ribbon indicating the flow behavior of that sector (Figure 2.27a). In more complex situations we fill elliptic sectors with a greenish semi-transparent surface and leave hyperbolic sectors open (Figure 2.27b). In the end it turns out, that applying this visualization scheme to first order critical points yields icons similar to the ones introduced in section 2.2.1 (Figure 2.27c).

Chapter 3

Concepts and Basic Algorithms for Extraction

In the previous chapter we discussed the topology of vector fields and identified a number of characteristic features suitable for insightful visualizations. A major goal of this thesis is to give algorithms for extracting and tracking these features – this will be addressed in detail in the following chapters. This chapter supports this goal by introducing a small but comprehensive set of tools as a foundation of these feature extraction algorithms. Most of our extraction techniques will be based upon the concepts formulated in this chapter. In general, our approach to feature extraction follows these principles:

- **generic concepts**
  The description of different features using the same generic concept allows to treat these different features in a similar way. Beside a similar mathematical treatment, this refers to their extraction using similar or even the same algorithms. Generic concepts are the theoretical basis for reusing algorithms.

- **reusability of algorithms**
  By reusing algorithms to extract different features we reduce the implementational efforts needed for a complete feature extraction library. It allows to concentrate on a smaller set of algorithms and use implementations which have already been proven to be stable and to produce valid results in other contexts.

- **chaining of algorithms**
  The output of an extraction algorithm should be usable as an input of another algorithm – allowing to build complex extraction techniques by chaining basic algorithms.

- **grid independence**
  In order to maximize the applicability, the extraction techniques should be independent of the underlying grid as much as possible. This can be achieved by treating the data as an interpolated field, i.e., one has to implement an interpolation function for each type of grid, but all extraction techniques utilize the interpolated, continuous field only.

The design of the feature extraction techniques developed in the course of this thesis has been guided by these principles in order to achieve a broad applicability. As an example, in section 5.1.1 we show how critical points can be tracked regardless of the underlying grid and even if the grid is changing over time. However, such generic approaches are usually not as fast as algorithms which have been specifically designed for a certain type of data, e.g., Garth et al. [GTS04b] give a fast algorithm for tracking critical points in tetrahedral grids where the grid needs to remain constant over time.

In section 3.1 we discuss the concept of Feature Flow Fields (FFF) which have been introduced by Theisel and Seidel in [TS03]. This approach is mainly used to track features in a time-dependent vector field **v** by introducing an appropriate vector field **f** in space-time, such that a feature tracking in **v** corresponds to a stream line integration in **f**.

The original approach of feature tracking using FFF requested that the complete vector field **v** is kept in main memory. Especially for 3D vector fields this may be a serious restriction, since the size of time-dependent vector fields can exceed the main memory of even high-end workstations. In the course of this thesis we developed a modification of the FFF-based tracking approach which works in an out-of-core manner (section 3.1.1). For an important subclass of all possible FFF-based tracking algorithms we ensure to analyze the data in one sweep while holding only two consecutive time steps in main memory at once. Similar to the original approach, the new modification guarantees the complete feature skeleton to be found. The theoretical findings of section 3.1.1 will be applied in section 5.1.1 to the tracking of critical points in 2D and 3D time-dependent vector fields.

Another important generic concept for feature extraction is the Parallel Vectors (PV) operator introduced by Peikert and Roth [PR99]. It is often used to extract core lines of swirling motion in flow fields, but it is also applicable to ridge lines in scalar fields and a number of other features. In section 3.1.2 we show how every PV problem can be formulated using FFF.

Section 3.2 deals with the concept of connectors – an approach developed in the course of this thesis. Connectors are stream objects of a vector field **v** between two arbitrary structures (points, lines, ...). We use them to describe global features of **v**.

In section 3.3 we outline a software architecture aimed at the extraction and tracking of a variety of features in scalar, vector and tensor fields. The so-called Unified Feature Extraction Architecture has been developed in the course of this thesis and consists of three core algorithms only, namely *finding zeros*, *integrating* and *intersecting stream objects*. Following the principles above, these core algorithms can be combined in different ways in order to build complex extraction techniques. In fact, most of the extraction techniques in later chapters build upon this architecture.

3.1 Feature Flow Fields

The concept of feature flow fields (FFF) was first introduced by Theisel and Seidel in [TS03]. It follows a rather generic idea:

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

FFF have been used for a number of applications:

- Tracking critical points in time-dependent vector fields [TS03, TWHS04b, TWHS05],
- Topological simplification based on critical point tracking [TRS03a],

- Comparison of vector fields based on critical point tracking [TRS03c],
- Tracking closed stream lines [TWHS04b, TWHS05],
- Extraction of vortex core lines defined as ridges/valleys of Galilean invariant quantities [SWH05b],
- Extraction and tracking of vortex core lines defined as centers of swirling motion [TSW$^+$05],
- Extraction of topological lines in tensor fields [ZP04, ZPP05],
- Identification of periodic phenomena from insufficiently time-resolved data sets measured using PIV [DLBB07].

Furthermore, in some of these applications the derived feature flow fields are not only used to extract or track features, but also to localize, characterize and classify bifurcations in this context. Hence, FFF give an important theoretical tool, too.

Feature flow fields are commonly used with local features, i.e., features which can be described by a local analysis of the underlying field and possibly its derivatives. Here, **f** can usually be described by an explicit formula. In contrast to this, a FFF for a global feature can only be given in an implicit manner, since it can neither be decided locally whether a point belongs to a feature nor into which direction the feature evolves. Instead, the FFF approach has to be tightly coupled with a global feature detection strategy in order to assess global features. A suitable global strategy is the Connectors approach which will be explained in section 3.2. An example for an extraction algorithm based on FFF and Connectors is the tracking of closed stream lines (section 5.1).

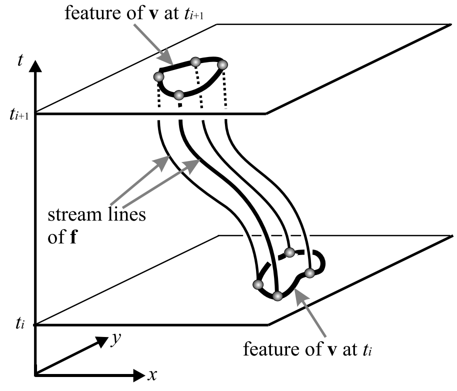
Tracking features in time-dependent fields is one of the main applications of feature flow fields and it shall be discussed in more detail. In a time-dependent field **v**, **f** describes the dynamic behavior of the features of **v**: for a field **v** with n spatial dimensions, **f** is a vector field $\mathbb{E}^{n+1} \to \mathbb{R}^{n+1}$. The temporal evolution of the features of **v** is described by the stream lines of **f**. In fact, tracking features over time is now carried out by tracing stream lines. The location of a feature at a certain time t_i can be obtained by intersecting the stream lines with the time plane t_i. Figure 3.1a gives an illustration.

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

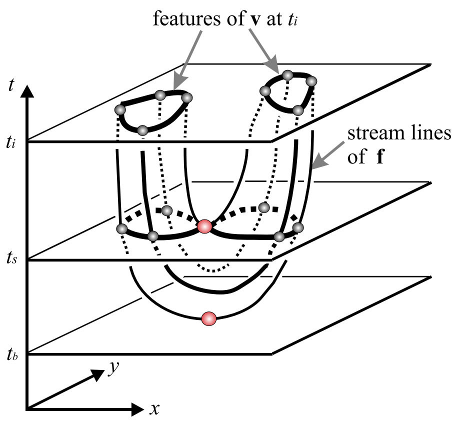
- A birth event occurs at a time t_b, if the feature at this time is only described by one point of a stream object of **f**, and all stream lines in the neighborhood of this point are in the half-space $t \geq t_b$.
- A split occurs at a time t_s, if one of the stream lines of **f** describing the feature touches the plane $t = t_s$ “from above”.
- An exit event occurs if all stream lines of **f** describing the feature leave the spatial domain.

The conditions for the reverse events (death, merge, entry) can be formulated in a similar way. Figure 3.1b illustrates the different events.

Integrating the stream lines of **f** in forward direction does not necessarily mean to move forward in time. In general, those directions are unrelated. The direction in time



(a) Tracking features by tracing stream lines.



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

Figure 3.1: Feature tracking using feature flow fields. Features at t_{i+1} can be observed by intersecting these stream lines with the time plane $t = t_{i+1}$.

may even change along the same stream line as it is shown in figure 3.1b. This situation is always linked to either a birth and a split event, or a merge and a death event.

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

Algorithm 1 *General FFF-based tracking*

1. *Get seeding points/lines/structures such that the stream object integration of* **f** *guarantees to cover all paths of all features of* **v**.
2. *From the seeding structures: apply a numerical stream object integration of* **f** *in both forward and/or backward direction until it leaves the space-time domain.*
3. *If necessary: remove multiply integrated stream objects.*

Algorithm 1 is more or less an abstract template for a specific FFF-based tracking algorithm like e.g. tracking of critical points. However, it shows a vital contradiction to out-of-core data handling: it gives no guarantee on how the data is processed. We would end up loading different data slices more than once, since both forward and backward integration of **f** are allowed, and as already said, the direction in time may even change along the same stream line.

3.1.1 Feature Flow Fields in Out-Of-Core Settings

The resolution of numerical simulations as well as experimental measurements like PIV have evolved significantly in the last years. The challenge of understanding the intricate flow structures within their massive result data sets has made automatic feature extraction schemes popular. Feature-based analysis can be seen as a kind of data reduction since it brings the raw data mass down to a small number of graphical primitives that ought to give insight into the flow structures. While the outcome of most feature extraction algorithms has a rather small memory footprint, the input often exceeds the main memory of high-end workstations. This is especially true for 3D time-dependent data. Thus, feature extraction algorithms should be compatible to an out-of-core data handling, i.e., treating only a small part of the input at once.

A number of algorithms already work in an out-of-core manner. Tricoche et al. [TWSH02] and Garth et al. [GTS04b] show how to track critical points in piecewise linear vector fields by analyzing the data in one sweep and holding only two time slices at once. Their approaches exploit the linearity of non-changing piecewise linear grids and are probably the best way to go for this important class of data.

Another way of tracking features which are defined by the parallel vector operator [PR99] is introduced in [BP02]. This approach is based on a 4-dimensional isosurface extraction – and therefore compatible to out-of-core data handling.

In this section we want to answer the question whether feature flow fields are compatible to out-of-core data handling, too. As already said, algorithm 1 fails to have that property. As reason for this could be considered, that the FFF approach is entirely based on the description of the data as a *continuous* field. At first glance, this concept seems to contradict the principle of out-of-core data handling: treating only a small part of the data at once. In this section we show that those two concepts do *not* contradict. In fact, we show how all FFF-based tracking algorithms can be formulated in an out-of-core manner. The theoretical results from this section will be used in section 5.1.1 to re-formulate the algorithm for tracking critical points from [TS03] to make it compatible to out-of-core data handling. The resulting algorithm enables to analyze the data in one sweep while holding only two time slices at once.

Before we formulate algorithm 1 in an out-of-core manner, we review out-of-core data handling in general and collect some concepts and properties of the FFF integration on which the new algorithm is based upon.

Out-Of-Core Data Handling

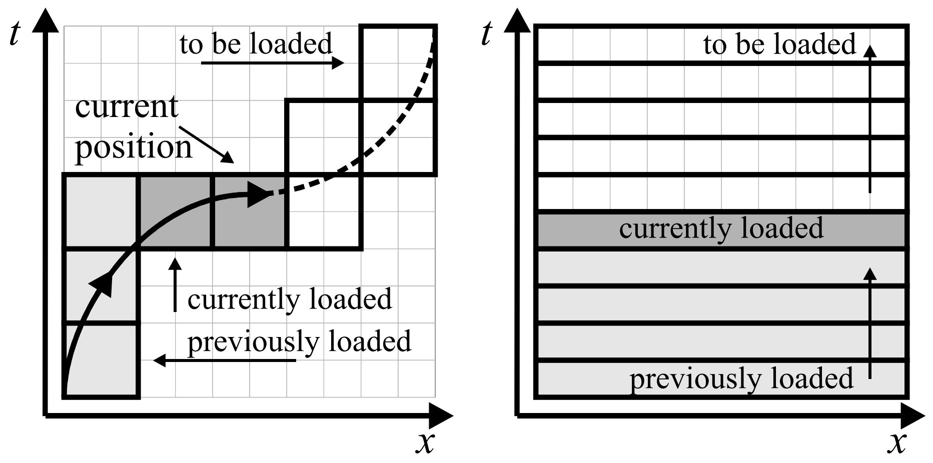
Out-of-core refers to the data handling strategy of algorithms, which process data too large to fit into main memory. Thus, only parts of the data can be loaded at once and acted upon. Since loading the data from a mass storage device is very time-consuming, the number of those operations should be reduced to a minimum. This restriction must already be considered when formulating the algorithm.

There are different types of out-of-core data handling strategies. We just want to mention two here:

- *Block-wise random access:* Data is loaded in blocks of same size. All dimensions are treated equally. The loaded data block with the oldest access time is subject to be substituted with the next block to be loaded. An application for this access pattern is the integration of a path line, which touches only parts of the domain. Figure 3.2a gives an illustration.

- *Slice-wise sequential access:* Data is loaded in slices, i.e., one dimension is fixed for every slice. Slices will be loaded as a fixed sequence in ascending or descending order. The procedure to load all slices from first to last is called a *sweep*. A very common approach is the usage of time slices, since a number of data sets are organized such that each time step is given as a separate file. An application for this access pattern is the extraction of fold bifurcations, where all parts of the domain need to be examined. Figure 3.2b gives an illustration.

Feature extraction algorithms usually do not have a-priori knowledge about the location of the feature and therefore, they need to examine the whole domain. A slice-wise



(a) Block-wise random access. (b) Slice-wise sequential access.

Figure 3.2: Out-of-core data handling strategies.

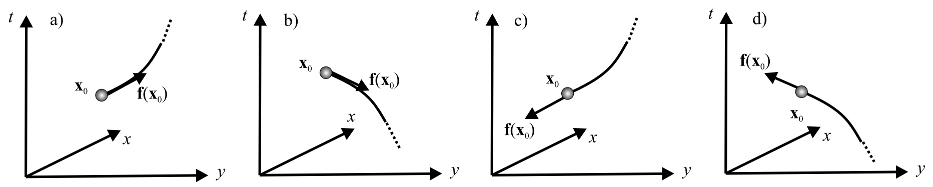

t
a)
$\mathbf{x}_0$
$\mathbf{f}(\mathbf{x}_0)$
x
y
t
b)
$\mathbf{x}_0$
$\mathbf{f}(\mathbf{x}_0)$
x
y
t
c)
$\mathbf{x}_0$
$\mathbf{f}(\mathbf{x}_0)$
x
y
t
d)
$\mathbf{f}(\mathbf{x}_0)$
$\mathbf{x}_0$
x
y

Figure 3.3: Orientation of integrating **f**: a) forward and t-forward; b) forward and t-backward; c) backward and t-forward; d) backward and t-backward; the curves are the integrated stream lines starting from $\mathbf{x}_0$.

sequential access strategy seems to be even more preferable, if the data is already given in time slices. For the rest of this thesis we consider this data handling strategy only. Note, that by loading two consecutive time steps t_i and t_{i+1} and applying a linear interpolation in between them, we obtain the time-dependent field in that time interval.

Direction of Integration Regarding Time

The FFF approach is based on a stream line integration of **f**. Given a starting point $\mathbf{x}_0 = (\mathbf{x}_0^s, t_0)$, **f** can be integrated in forward or backward direction. Assuming an Euler integration[1], the forward integration goes to the next point $\mathbf{x}_1 = \mathbf{x}_0 + \varepsilon\,\mathbf{f}(\mathbf{x}_0)$, while the backward integration gives the next point $\mathbf{x}_1 = \mathbf{x}_0 - \varepsilon\,\mathbf{f}(\mathbf{x}_0)$ for a certain small positive ε. In addition to this distinction of the integration orientation, we can also distinguish a t-forward and t-backward orientation. We call an integration *t-forward* if the next point $\mathbf{x}_1 = (x_1^s, t_1)$ is ahead in time, i.e., if $t_1 > t_0$. For $t_1 < t_0$, we have a *t-backward* integration. This property can be decided locally for a point $\mathbf{x}_0$ by looking at the sign of the t-component of $\mathbf{f}(\mathbf{x}_0)$, or if this component is zero, by looking at the sign of the partial derivative $\mathbf{f}_t(\mathbf{x}_0)$. Figure 3.3 illustrates some of these cases.

[1]For the actual integration we used a fourth order Runge Kutta method, the Euler integration is only for explaining the concepts.

Classification of Seeding Structures

The FFF approach is also based on finding appropriate starting structures for the integration. The definition of a complete set of seeding structures is up to the specific FFF-based application. However, we can give the following classification of those structures:

- ***t*-forward structures:** all integrations started here are t-forward only.
- ***t*-backward structures:** all integrations started here are t-backward only.
- **intermediate structures:** a t-forward and a t-backward integration will be started here.

This classification is independent of a specific FFF-based application, though it might be that in certain cases a class of structures is empty, e.g. there are only t-forward and intermediate structures but no t-backward structures.

As already discussed, a t-forward integration may change to a t-backward integration even along the same stream line. This situation is always linked to either a birth or a death event, which perfectly fit into the classification: a birth event is a t-forward point, and a death event a t-backward point.

Out-Of-Core FFF-based Tracking Algorithm

The split of the integration into different directions regarding the time is the conceptual key to an out-of-core version of algorithm 1:

Algorithm 2 *Out-of-core FFF-based tracking*

1. *Load the data in a slice-wise sequential manner from t_{min} to t_{max}. For each time interval between the time slices t_i and t_{i+1}:*
   (a) *Get seeding structures such that the stream object integration of* $\mathbf{f}$ *guarantees to cover all paths of all features of* $\mathbf{v}$.
   (b) *Classify the seeding structures into t-forward, t-backward and intermediate structures.*
   (c) *Start a t-forward integration at all t-forward and intermediate structures. Stop the integration, if*
      i. *the spatial domain was left.*
      ii. *the temporal domain of the time interval was left, i.e., the integration reached t_{i+1}. Add the result of this integration to the list of t-forward structures for the next time interval.*
      iii. *a death event was reached. If this point will not be reached by any other t-forward integration, add the result to the list of t-backward structures.*
2. *If the list of t-backward structures is non-empty: load the data in a slice-wise sequential manner from t_{max} to t_{min}. For each time interval between the time slices t_i and t_{i-1} start a t-backward integration at all t-backward and intermediate structures similar as above.*

3. *Repeat the stream object integrations of steps 1 and 2 until the lists of seeding structures are empty.*

The basic idea of this algorithm template is to ensure that the direction of loading the data coincides with the direction of integration, i.e., if we are loading the data from t_{min} to t_{max} then we are integrating t-forward only, and the other way around. This alone does not sound very effective, since the data might been loaded more than once, possibly even an unknown number of times.

This changes, if we take a closer look at the t-forward structures, i.e., all points where *only* a t-forward integration is intended. At those points the features appear for the first time. Examples are entry points, birth events or all occurrences of the feature at t_{min}. If we can find all t-forward structures while doing the first sweep through the data, then the whole feature skeleton can be extracted with this one sweep. This is always fulfilled, if all types of t-forward structures are locally defined, i.e., they can be extracted by a local analysis. Under these prerequisites, we can reformulate algorithm 2 and obtain the following algorithmic template:

Algorithm 3 *One-sweep Out-of-core FFF-based tracking*

```
1. For each time interval $[t_i, t_{i+1}]$ from $t_{min}$ to $t_{max}$:
    (a) Extract all t-forward seeding structures needed to cover all paths of all
        features of $\mathbf{v}$.
    (b) Apply a t-forward integration starting at those structures until
        i. the spatial domain was left.
        ii. the temporal domain of the time interval was left, i.e., the integration
            reached $t_{i+1}$. Add the result of this integration to the list of t-forward
            structures for the next time interval.
        iii. a death event was reached.
```

Algorithm 3 ensures that every path of a feature is integrated only once. Thus, a removal of multiple integrated stream objects is not needed anymore. In comparison to algorithms 1 and 2 it is perfectly fitted for large data sets: it reads only parts of the data and each part only once.

Based on this, we will formulate an out-of-core compatible algorithm for tracking critical points in section 5.1.1.

3.1.2 Feature Flow Fields and the Parallel Vectors Operator

An important concept for feature extraction is the Parallel Vectors (PV) operator introduced by Peikert and Roth [PR99]. A number of features like core lines of swirling motion in flow fields or extremal lines in scalar fields can be formulated using this operator [PR99, BP02, TSW$^+$05, WSTH07]. Using a single concept to define a variety of features allows reuse of algorithms and their implementations.

In this section we show that every PV problem can be reformulated using the concept of feature flow fields. This allows to use an alternative extraction scheme for all features defined by the PV operator. While the original extraction algorithm is cell based [PR99], the FFF approach is grid independent due to the utilization of stream line integration. It was already stated in [ZPP05] that the FFF approach leads to shorter computation times than cell based algorithms in situations where a resolution is desired which is much finer than the original grid.

The PV operator can be described as follows: given two continuous 3D vector fields $\mathbf{w}_1$ and $\mathbf{w}_2$, the PV operator extracts all points in the domain where the vectors of $\mathbf{w}_1$ and $\mathbf{w}_2$ are parallel, i.e., $\mathbf{w}_2 = \lambda \mathbf{w}_1$ for some real λ, or $\mathbf{w}_1 \parallel \mathbf{w}_2$ in shorthand notation.[2] Particular choices of $\mathbf{w}_1$ and $\mathbf{w}_2$ depend on the application. Examples can be found in [PR99, TSW$^+$05].

Aiming at extracting the loci of all points where $\mathbf{w}_1 \parallel \mathbf{w}_2$, we define the vector field $\mathbf{s}$ as

$$\mathbf{s}(\mathbf{x}) = \mathbf{w}_1 \times \mathbf{w}_2. \tag{3.1}$$

Hence, all locations with $\mathbf{w}_1 \parallel \mathbf{w}_2$ are now described by $\mathbf{s}(\mathbf{x}) = (0,0,0)^T$. If $\mathbf{w}_1$ and $\mathbf{w}_2$ are continuous, then this gives continuous line structures, i.e., point sets of dimensionality 1 [PR99].[3] PV structures of dimensionality 0 or dimensionality 2 are structurally unstable in 3D, i.e., they disappear by adding noise to the data. For this reason we do not consider them here.

In order to describe PV lines using FFF, the following steps are necessary:

1. A vector field $\mathbf{f}$ is defined which fulfills the FFF property: PV lines of ($\mathbf{w}_1$, $\mathbf{w}_2$) are stream lines of $\mathbf{f}$. This means that for a given point $\mathbf{x}_0$ with $\mathbf{s}(\mathbf{x}_0) = (0,0,0)^T$, each point $\mathbf{x}$ on the stream line of $\mathbf{f}$ through $\mathbf{x}_0$ fulfills $\mathbf{s}(\mathbf{x}) = (0,0,0)^T$, too.

2. A set of starting points is defined which guarantees that the stream line integration of $\mathbf{f}$ starting from them covers all PV lines.

Then all PV lines of ($\mathbf{w}_1$, $\mathbf{w}_2$) can simply be extracted by applying a stream line integration of $\mathbf{f}$. It can be shown that the desired feature flow field is given by

$$\mathbf{f} = \begin{pmatrix} \det(\mathbf{s}_y, \mathbf{s}_z, \mathbf{a}) \\ \det(\mathbf{s}_z, \mathbf{s}_x, \mathbf{a}) \\ \det(\mathbf{s}_x, \mathbf{s}_y, \mathbf{a}) \end{pmatrix} \tag{3.2}$$

where $\mathbf{a}$ is an almost arbitrary auxiliary vector field with the only restriction that it must not be perpendicular to $\mathbf{w}_1$ and $\mathbf{w}_2$ respectively on a PV line. If we know that $\mathbf{w}_1$ never vanishes on a PV line, the simple choice $\mathbf{a} = \mathbf{w}_1$ does the job. A similar statement holds for $\mathbf{w}_2$. In case that both $\mathbf{w}_1$ and $\mathbf{w}_2$ may vanish on a PV line, we choose

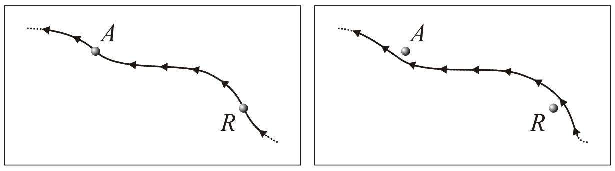
$$\mathbf{a} = \begin{cases} \mathbf{w}_1, & \text{if } \|\mathbf{w}_1\| \geq \|\mathbf{w}_2\| \\ \mathbf{w}_2, & \text{otherwise} \end{cases}$$

which guarantees $\mathbf{a}$ to be continuous in direction but not in orientation. Thus, $\mathbf{f}$ has to be integrated as an orientation-free vector field (similar e.g. to an eigenvector field of a tensor field), i.e., the local orientation has to be obtained from the information where the integration of the line has come from.

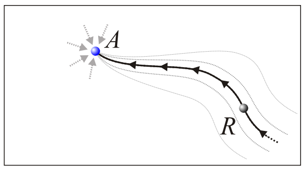
A suitable set of starting points can be found by taking into account that every PV line either ends on the boundary of the domain or is a closed line [PR99]. In the first case, we search for the intersections of the PV lines with the boundary, i.e., all isolated

[2] $\lambda = \pm\infty$ is also allowed, i.e., $\mathbf{w}_1 \parallel \mathbf{w}_2$ holds if $\mathbf{w}_1$ or $\mathbf{w}_2$ vanishes.

[3] Note that this statement gives that these line structures cannot be obtained by replacing $\mathbf{s}(\mathbf{x}) = (0,0,0)^T$ with $\|\mathbf{s}(\mathbf{x})\| = 0$ and applying a simple scalar field analysis of $\|\mathbf{s}\|$, since the zeros of general scalar fields are structures of dimensionality 2, i.e., isosurfaces.



(a) Connector between two points A and R as a stream line (left). The connector is unstable since it disappears by adding a small perturbation to the field (right).



(b) A stable connector between two points A and R where A is a sink.

Figure 3.4: Stable and unstable connectors between two points.

points on the boundary with $\mathbf{s} = 0$. This is equivalent to finding critical points of a 2D vector field. A starting point on a closed PV line can be found by requiring that one of the components of $\mathbf{f}$ is zero in addition to $\mathbf{s} = 0$. This is equivalent to finding critical points of a 3D vector field.

A more detailed analysis of the relation between FFF and PV can be found in [TSW$^+$05].

3.2 Connectors

The concept of connectors has been developed in the course of this thesis. Since it involves the integration of stream objects, it is an inherently global approach:

Consider two arbitrary structures A and R (points, lines, surfaces, ...) in a vector field $\mathbf{v}$. A *connector* is a stream object which passes through both A and R. Considering a forward integration of $\mathbf{v}$, the connector passes through R first and reaches A afterwards. The structures A and R may have an attracting and repelling behavior in the flow field, i.e., the connector may as well *start* in R and *end* in A.

The concept of connectors has been used in a number of applications:

- Extraction of saddle connectors [TWHS03],
- Extraction of boundary switch connectors [WTHS04a],
- Extraction of closed stream lines [TWHS04a],
- Tracking closed stream lines [TWHS04b, TWHS05],
- Extraction of global events like saddle connections, periodic blue sky bifurcations, cyclic fold bifurcations [TWHS04b, TWHS05].

Figure 3.4a shows a connector which first passes through the point R and subsequently through point A. However, given that R and A are non-critical and their positions are fixed, such a connector is structurally unstable. This means that the connector disappears by adding a small perturbation to the vector field. A stable connector between two points R and A is given if R is a source and/or A is a sink. Figure 3.4b illustrates this.

An important subclass of connectors can be described as the intersection of two stream objects S_R and S_A, where S_R emanates from R in forward integration and S_A emanates from A in backward integration. Given a n-dimensional vector field $\mathbf{v}$, an

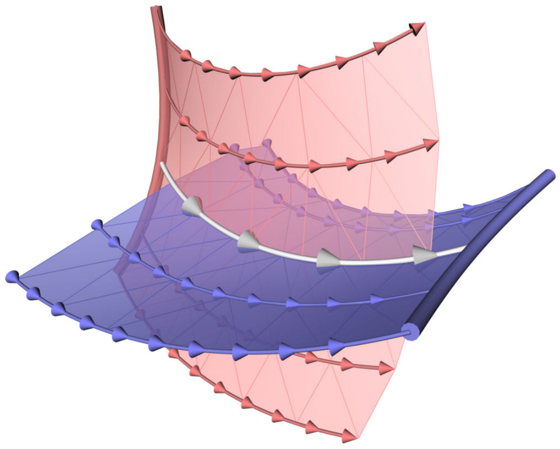


Figure 3.5: Connector as the intersection of two stream objects. Two stream surfaces S_A (blue) and S_R (red) emanate from their respective seeding curves A and R in backward/forward integration. Their intersection is a stream line (gray) – a structurally stable connector between the seeding curves.

intersection of two stream objects S_R and S_A of dimensionality $n-1$ is always a structurally stable connector of dimensionality $n-2$ from R to A.[4] Figure 3.5 illustrates this for a 3D vector field where A and R are two seeding curves giving rise to the stream surfaces S_A and S_R respectively. The intersection of these surfaces is a stream line, a structurally stable connector between the seeding curves.

Further considerations are restricted to connectors which can be described as intersections of stream surfaces in 3D vector fields. Section 3.3.3 gives an algorithm for extracting such connectors.

3.3 Unified Feature Extraction Architecture

In the course of this thesis we have developed a software architecture which allows to extract and track a rich variety of topological and other features. The goal is to follow the principles discussed at the beginning of this chapter, i.e., we want to use generic concepts in order to reuse and chain grid independent algorithms. This effort aims at a broad applicability while alleviating the implementational expenses.

In fact, only three algorithms have to be implemented, since almost every feature can be extracted and tracked using a combination of the following core algorithms:

- Finding Zeros
- Integrating Stream Objects
- Intersecting Stream Objects

The earlier in this chapter discussed concepts are embodied in this: Feature Flow Fields can be expressed using the first two algorithms and Connectors using the second two. Figure 3.6 gives an overview of the Unified Feature Extraction Architecture (UFEA).

In order to build more complex feature extraction techniques from these simple three core algorithms, the output of one algorithm has to become the input of another one. This can easily be seen: the result of *Finding Zeros* are isolated points which can serve as seeding points of a stream line integration. Such lines can be used as seeding curves for a stream surface integration. Certain stream surfaces can intersect each other and the result is a set of stream lines again. Hence, we can derive a number

[4]We exclude the unstable case of partially collapsing stream objects S_R and S_A.

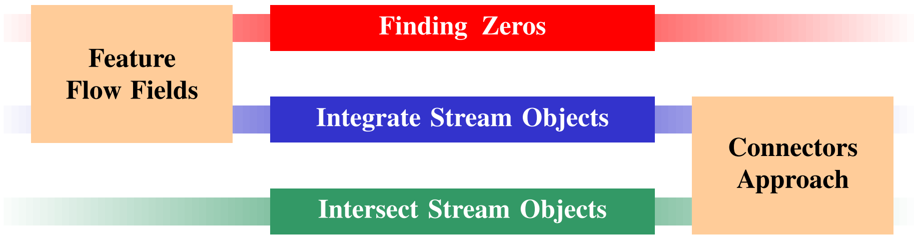


Figure 3.6: Unified Feature Extraction Architecture. In order to extract and track a variety of features it suffices to implement three core algorithms: *finding zeros*, *integrating* and *intersecting stream objects*. A vital part of the architecture are the concepts of Feature Flow Fields and Connectors, which can be expressed using these three algorithms.

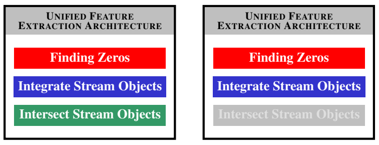


Figure 3.7: Icons for indicating which UFEA-algorithms have been used to extract a specific feature (denoted by color).

of extraction techniques as a combination of the three core algorithms. In fact, most of the feature extraction techniques presented in later chapters have been built that way. At these places we will indicate the utilized algorithms by a small icon similar to the ones shown in figure 3.7.

It is not the goal of this thesis to define a mathematically rigorous set of features that can be handled using UFEA nor to theorize the interplay between the core algorithms. In contrast to other parts of the thesis, this is meant to be rather informal, but still useful to describe the underlying software on a higher abstraction level and it hopefully serves as a starting ground to make the implementation of a feature extraction library easier than it was at the beginning of this work. An implementation of such a library based on this architecture needs to have access to data interpolation, derivation and possibly out-of-core data handling. These are topics of their own and will not be treated here. However, most visualization packages have interpolation, derivation etc. already built in.

To the best of our knowledge, there has been only one generic approach to feature extraction prior to this work, namely the PV operator [PR99] as briefly discussed in section 3.1.2. Peikert and Roth have given a single algorithm to extract all the different features described by this operator. However, only local features can be treated this way, while important topological features like separatrices or saddle connectors are left out since they are global. Furthermore, we have shown in section 3.1.2 that every PV problem can be reformulated using FFF, i.e., all features described by the PV operator can be extracted using our architecture as well. To the best of our knowledge, the UFEA is the first generic approach to feature extraction treating local and global features.

In the following sections we will comment on each of the three core algorithms. There exist a number of solutions for the first two problems in the literature – therefore we discuss the finding of zeros and the integration of stream objects only briefly here. The problem of intersecting stream objects has been solved in the course of this thesis and will be covered more extensively.

3.3.1 Finding Zeros

We aim at extracting isolated zeros of scalar, vector and tensor fields. Generally spoken, we want to extract all isolated locations where the values of a field become zero.

Several approaches exist for this matter. One might use a Newton-Raphson approach where the first derivative is used to repeatedly predict a zero (e.g. applied in [PR99]). If the field is interpolated from grid data, one can utilize this knowledge. For example, in piecewise linear fields like tetrahedral grids, the zeros can be computed explicitly. This has been used in [GTS04b] to extract critical points of time-dependent flow fields.

Mann et al. [MR02] use an octree-like approach to find critical points. After dividing the domain into uniform cells, they compute the index of each cell. A non-zero index is a sufficient condition for the existence of at least one critical point in that cell. In order to locate the critical points, a recursive subdivision will be applied to each cell with non-zero index. However, this criterion is not a necessary condition: a cell with an index of zero may as well have critical points inside, where the sum of their indices is zero. Hence, the resolution of the initial domain division is of great importance to the effectiveness of this approach. Note, that this algorithm works for non-regular grids as well by sampling the field onto the uniform grid in the first stage. Since this algorithm relies on the index, this works for vector fields only and cannot be extended to scalar or tensor fields.

An octree-like approach specifically designed for electric fields defined by a set of point charges has been presented in [MW07]. The given analytical criterion excludes all cells from a subdivision where critical points will not appear. Hence, critical points cannot be missed. The algorithm can only be applied to this specific kind of vector fields.

The mainly used algorithm in our framework resembles the idea of [MR02], but uses a different criterion to decide about the subdivision in order to treat scalar and tensor fields as well: we check whether at least one component of the field is positive/negative at *all* corners of a cell. If so, a zero cannot appear in that cell. Otherwise, we recursively subdivide the cell until a certain threshold is reached. This component-wise change-of-sign test is a necessary condition for a zero inside a (bi-/tri-)linearly interpolated grid cell. While this is less powerful than the index criterion and therefore possibly imposes a higher resolution of the initial domain division, the change-of-sign test is much faster to compute and a number of real word data sets are already given as piecewise trilinear fields on regular grids: here, our criterion can readily be applied to the original grid and zeros cannot be missed. Furthermore, our criterion is easy to implement and since it applies to scalar, vector and tensor fields in all dimensions, we favor it over other methods.

3.3.2 Integrating Stream Objects

In section 2.1.2 we discussed integral curves of vector fields and have shown that stream and path lines can be computed as tangent curves of certain autonomous ODE systems (2.9), (2.13), (2.14), (2.15). A stream object is a set of integral curves. Its dimension depends on the dimensionality of the seeding structure, i.e., a m-dimensional seeding structure gives rise to a $m+1$-dimensional stream object:

- from a seeding point $\mathbf{p}$ ($m = 0$) we obtain a tangent curve (which is a stream or path line depending on the underlying ODE system),

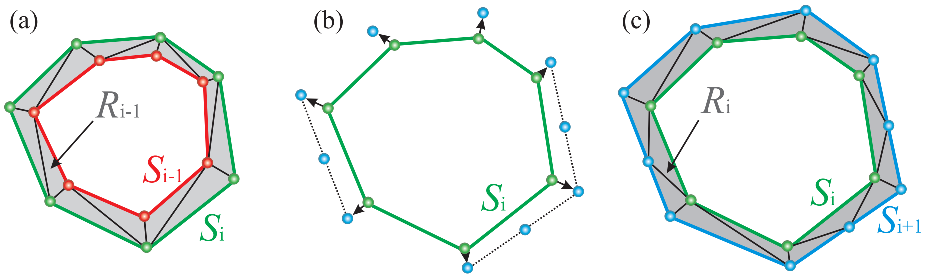


Figure 3.8: Computing time ribbons of a stream surface; (a) the time line S_{i-1} consists of the red points and edges, the time line S_i is colored green; the time ribbon R_{i-1} is described by the triangulation between S_{i-1} and S_i (gray area); (b) to compute the time ribbon R_i, we apply one numerical integration step to each point of S_i and adaptively insert/remove points; this way we obtain S_{i+1} (blue points); (c) the new time ribbon R_i is the triangulation between S_i and S_{i+1} (gray area).

- from a seeding curve S ($m = 1$) we obtain a stream surface consisting of all tangent curves with the seeding points $\mathbf{p} \in S$,
- from a seeding surface A ($m = 2$) we obtain a flow volume consisting of all tangent curves with the seeding points $\mathbf{p} \in A$.

In order to integrate a single tangent curve we use a fourth-order Runge-Kutta integration with error monitoring and adaptive step size control (see e.g. [SH95a]). The computation of stream surfaces and flow volumes can be reduced to the integration of a finite number of stream lines and appropriate triangulation/tetrahedrization between them. Max et al. give such an algorithm for flow volumes in [MBC93]. Flow volumes will not be considered in this thesis.

In order to compute stream surfaces we use a technique similar to the one proposed by Hultquist in [Hul92]. First, we place n_0 points on the seeding curve. They serve as starting points for the stream lines approximating the stream surface. The seeding curve is denoted as time line S_0. To get the time line S_{i+1}, we apply one step of a numerical stream line integration to all points of $S_i = (\mathbf{s}_{0,i}, \ldots, \mathbf{s}_{n_i,i})$. To the n_i new points obtained this way, an adaptive thinning or enrichment is applied in order to maintain a given resolution of the approximated stream surface. Different methods exist for this [Hul92, Sta98, GTS$^+$04a]. We use the method of Stalling [Sta98]. This results in a new number of points for $S_{i+1} = (\mathbf{s}_{0,i+1}, \ldots, \mathbf{s}_{n_{i+1},i+1})$. By applying a triangulation between S_i and S_{i+1}, we obtain a *time ribbon* R_i. Figure 3.8 gives an illustration.

For a number of applications it is beneficial to parameterize the surface such that every point on the surface can be mapped to the corresponding point on the seeding curve. To do so, for each stream line a parameter α is stored, which determines its starting point on the seeding curve. Hence, the seeding curve has to be parameterized accordingly. Such a parametrization can be used to construct a new point on the time line S_i. This works as follows: Consider two neighboring points $\mathbf{s}_{k,i}$ and $\mathbf{s}_{k+1,i}$ on the current time line S_i with their corresponding parameters α_k and α_{k+1}. Using these parameters we know their corresponding starting points $\mathbf{s}_{k,0}$ and $\mathbf{s}_{k+1,0}$ on the seeding curve. Furthermore, every $\alpha' \in (\alpha_k, \alpha_{k+1})$ gives a point on the seeding curve between $\mathbf{s}_{k,0}$ and $\mathbf{s}_{k+1,0}$. A stream line integration from such a point gives – after i time steps – a new point between $\mathbf{s}_{k,i}$ and $\mathbf{s}_{k+1,i}$ on the current time line S_i. This way of constructing

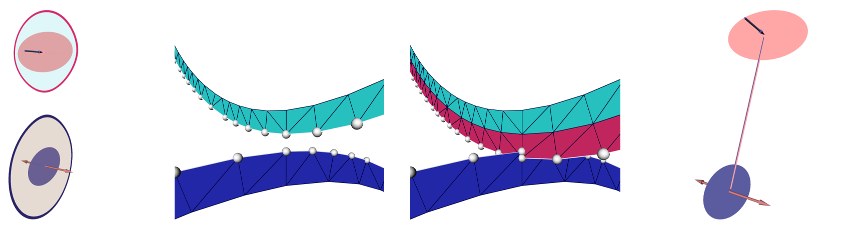

(a) Simultaneously compute the time ribbons. (b) Closeup shortly before an intersection is found. (c) Intersection is found. (d) Final connector.

Figure 3.9: Computing a connector.

a new point on S_i is used both for the adaptive enrichment described above and for the refinement step in the intersection algorithm described below.

3.3.3 Intersecting Stream Objects

This algorithm aims at finding connectors which are defined as the intersection of two stream objects. We consider the intersection of stream surfaces in 3D vector fields only. First, we give an algorithm for finding the intersection between two stream surfaces emanating from two seeding curves. Afterwards, we discuss how to find all connectors between a higher number of seeding curves.

Computing a Single Connector

Several approaches for finding connectors can be thought of. We want to discuss two of them here.

The first approach follows directly the definition of connectors as the intersection of two stream surfaces by integrating the stream surfaces and intersecting their triangle mesh representations. To ensure that the resulting intersection curves start and end at the seeding curves, each surface needs to be integrated until it comes very close to all other seeding curves that share a connector with the originating curve. Our experience has shown that especially for topologies with circulating flow behavior like focus saddles, this is hard to ensure. Furthermore, this approach does not only need a lot of integration steps but also a considerable amount of memory as one needs to hold all generated triangles.

We propose another algorithm with less memory consumption, which enables us to find connectors with less integration steps than the first approach. We utilize the fact that connectors are particular stream lines: knowing one (non–critical) point of a stream line is enough to integrate the whole line.

Consider a repelling seeding curve R and an attracting curve A. We apply a simultaneous integration of the stream surfaces until an intersection point $\mathbf{p}$ is found. This point lies close to the connector, but due to the triangle approximation of the surfaces it is not necessarily on the connector itself. Let $\mathbf{p}'$ be the point on the connector which is closest to $\mathbf{p}$. $\mathbf{p}'$ has two corresponding seeding points – one on each seeding curve. They can be described by the parameters $\alpha' \in (\alpha_k, \alpha_{k+1})$ and $\beta' \in (\beta_h, \beta_{h+1})$ and are found by applying a refinement.

To find $\mathbf{p}$ we only need to simultaneously compute the time ribbons R_i^R and R_i^A, and check them for intersection. For this it is sufficient to check if the last time line

Algorithm 4 Finding intersections between stream surfaces

```
$\mathbf{p}[\,] = \texttt{EmptyArrayOfPoints}()$
$S_0^A = \texttt{Seed}(A)$
$S_0^R = \texttt{Seed}(R)$

$S_1^A = \texttt{PropagateSurface}\ (A, S_0^A, t_1)$
$R_0^A = \texttt{Triangulate}\ (S_1^A, S_0^A)$

$i = 0$
while $(\texttt{Size}\ (\mathbf{p}[\,]) < n_{max}) \wedge (t_i < t_{max})$ do
    $S_{i+1}^R = \texttt{PropagateSurface}\ (R, S_i^R, t_{i+1})$
    $R_i^R = \texttt{Triangulate}\ (S_{i+1}^R, S_i^R)$

    $\mathbf{p}[\,] += \texttt{GetIntersectionPoints}\ (S_{i+1}^R, R_i^A)$
    $\mathbf{p}[\,] += \texttt{GetIntersectionPoints}\ (S_{i+1}^A, R_i^R)$

    $S_{i+2}^A = \texttt{PropagateSurface}\ (A, S_{i+1}^A, t_{i+2})$
    $R_{i+1}^A = \texttt{Triangulate}\ (S_{i+2}^A, S_{i+1}^A)$

    $\mathbf{p}[\,] += \texttt{GetIntersectionPoints}\ (S_{i+1}^R, R_{i+1}^A)$
    $\mathbf{p}[\,] += \texttt{GetIntersectionPoints}\ (S_{i+2}^A, R_i^R)$

    $\texttt{FreeMemory}\ (S_i^R, R_i^R, S_{i+1}^A, R_i^A)$
    $i = i + 1$
end while
```

of R intersects the last time ribbon of A and vice versa. This means that we do not have to check the triangular strips themselves for intersection, but line segments with triangular strips. Figure 3.9 illustrates the process of finding a connector and algorithm 4 gives a detailed description.

For two reasons this algorithm may find more than one intersection point. First, the seeding curves A and R may have multiple connectors (see figure 2.10b on page 29). Second, we perform two intersection tests after each surface propagation, and in many cases[5] both produce results at the same time step. The two points resulting from this are very close to each other, and one of them can be deleted safely. The algorithm terminates after finding a certain maximal number of intersection points (or if a maximal time is reached).

Algorithm 4 depends on a number of parameters: the step size of the stream line integration, the number of initial seed points, and the thresholds for inserting or deleting points for a new time line. For all these values, a reasonable compromise has to be found between performance and accuracy.

An advantage of algorithm 4 is that it avoids stream line integration in numerically complicated regions (e.g., when computing saddle connectors in regions in which the separation surface of a saddle comes close to another saddle and therefore has a strongly diverging behavior). Also, the algorithm stops after a certain pre-defined

[5]But not in all cases: If we use different step sizes for the integration of the surfaces, these two tests may result in only one intersection point. Thats why two tests are performed. Otherwise one time ribbon could “jump” over the other.

number n_{max} of maximal connectors is found. This leads to a significant reduction of necessary integration steps. Finally, note that memory is hardly a problem, since from each stream surface only a time ribbon (represented by a triangular strip) is stored at any stage of the algorithm.

Computing all Connectors

Consider more than two seeding curves, i.e., a set of attracting seeding curves and a set of repelling ones. This yields a number of forward and backward integrated stream surfaces and we want to find all intersections of them.

A naive approach to do so – applying algorithm 4 to any pair of repelling and attracting seeding curves – is not an appropriate solution, because the time-consuming integration of a stream surface is repeated again and again.

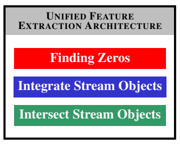
Instead, we avoid a repeated computation of a stream surface by grouping all attracting and all repelling seeding curves and applying a slightly modified version of algorithm 4 on these two groups: after computing a new time ribbon in one group, they have to be checked against all current time ribbons of the other group. If a time ribbon leaves the domain of the vector field or collapses to a number of single points, the time ribbon is excluded from further consideration. This algorithm has a time complexity of $O(n_A \times n_R)$ where n_A, n_R are the numbers of considered attracting/repelling seeding curves.

Chapter 4

Extraction of Parameter-Independent Topological Structures

In the following we give a number of algorithms for extracting topological features of parameter-independent vector fields. This includes techniques for computing saddle connectors (section 4.1) and boundary switch connectors (section 4.2), which have been newly developed in the course of this thesis. Furthermore, we present a new grid-independent algorithm for finding closed stream lines in planar flows (section 4.3). In section 4.4 we apply these algorithms to a number of data sets.

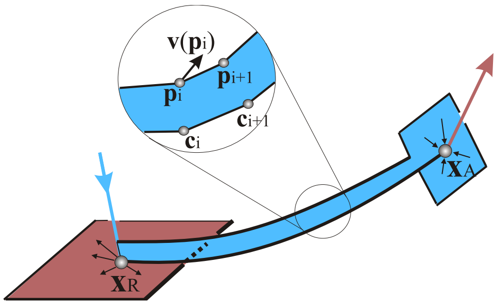
4.1 Features induced by Critical Points



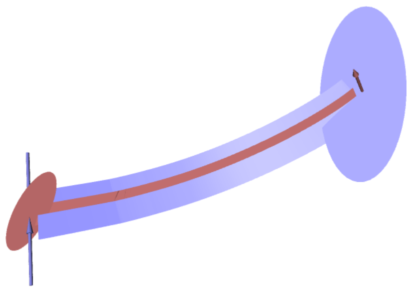
In a number of applications the boundary does not have a great influence on the flow and isolated closed stream lines cannot appear (divergence free flows). Here, it suffices to examine critical points and all topological features which are induced by them, i.e., the separatrices emanating from saddle points as well as saddle connectors. In the following we deal with the extraction of these features. We need all three algorithms of our Unified Feature Extraction Architecture for this. Furthermore, the concept of Connectors comes into play when we extract saddle connectors. Feature Flow Fields are not needed.

Critical points are the zeros of the considered vector field and can be extracted by simply applying the algorithm for finding zeros (see section 3.3.1). However, this yields only the locations of the critical points. In order to distinguish between sources, sinks and saddles we apply an eigenvalue/eigenvector analysis of the Jacobian matrix $\mathbf{J}$ and classify the points as described in 2.2.1. For the classification itself we need the eigenvalues only, but the depiction of the 3D critical points as icons requires the eigenvectors as well (see figure 2.5).

Eigenvectors are also needed to compute the separatrices starting from the saddle points. While their computation itself is done by applying the algorithm for stream object integration, the starting points of this integration have to be chosen carefully. Obviously, the saddle point itself cannot serve as a seeding point since it is critical



(a) Computing the flow ribbon of the connector between $\mathbf{x}_R$ and $\mathbf{x}_A$ belonging to the separation surface of $\mathbf{x}_A$.



(b) Double flow ribbon used for representing a saddle connector.

Figure 4.1: Visual representation of saddle connectors.

and an integration would be stuck. Instead we use the eigenvectors to find the seeding points close to the saddle and on the corresponding separatrix:

- For a separation line we take two small steps into the direction of its corresponding eigenvector - one step forward and one backward. This yields two seeding points for two stream line integrations.
- To integrate the separation surface of a saddle point $\mathbf{x}_0$, we place n_0 points in an equidistant manner on a small circle in the outflow/inflow plane around $\mathbf{x}_0$ spanned by the corresponding two eigenvectors. They serve as the seeding points of the separation surface. This closed polygon $S_0 = (\mathbf{s}_{0,0}, ..., \mathbf{s}_{n_0,0})$ can be considered as the time line of the separation surface for the time t_0. This kind of seeding yields good results for most topologies except for focus saddles with strong circulation, where the stream lines would intersect the seeding circle. In this case we place the seeding points on a small line in the outflow/inflow plane starting at the critical point.

The direction of the integration is determined by the type of the separatrix: attracting or repelling. Attracting separatrices have to be integrated in backward direction, repelling separatrices in forward direction. This can be found by the sign of the corresponding eigenvalue.

Visualizing the separation surfaces of topologically complex vector fields usually leads to visually cluttered images since the surfaces tend to hide each other as well as the critical points. A solution to this is the display of saddle connectors, which at least indicate the approximate run of the separation surfaces. In order to extract them we apply the algorithm for intersecting stream surfaces. Once saddle connectors are computed, we have to find a graphical representation. The simplest way is to use a line representation similar to stream lines. However, since saddle connectors are defined as the intersection of two separation surfaces, we can also visualize the orientation of the separation surfaces in the neighborhood of the saddle connector. To do so, we use a double flow ribbon approach as shown in figure 4.1b.

Given are the repelling saddle $\mathbf{x}_R$, the attracting saddle $\mathbf{x}_A$ and their saddle connector represented by a sequence $\mathbf{c}_0, \ldots, \mathbf{c}_m$ of points. To construct the flow ribbon belonging to the separation surface of $\mathbf{x}_A$, we have to find a sequence $\mathbf{p}_0, \ldots, \mathbf{p}_m$ of points in such a way that the distance between $\mathbf{c}_i$ and $\mathbf{p}_i$ is constant, and the line segments $(\mathbf{c}_i, \mathbf{p}_i)$ are in the separation surface of $\mathbf{x}_A$ for all i. To do so, we set $\mathbf{p}_0 = \mathbf{x}_R + \lambda \cdot \mathbf{e}$ where $\mathbf{e}$

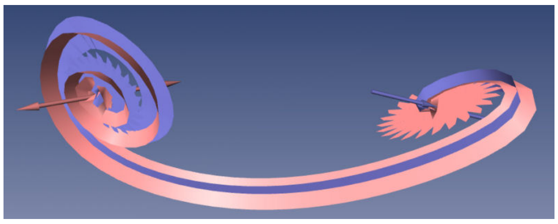


Figure 4.2: Flow behind a circular cylinder. Close-up of a saddle connector between two focus saddles.

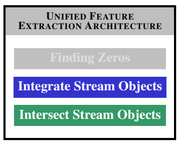
is the only eigenvector of $\mathbf{J_v}(\mathbf{x}_R)$ which has a negative eigenvalue, and λ is chosen in such a way that $\mathbf{c}_0$ and $\mathbf{p}_0$ have a pre-defined constant distance. Then the point $\mathbf{p}_{i+1}$ is constructed from $\mathbf{c}_i$, $\mathbf{c}_{i+1}$ and $\mathbf{p}_i$ by the following conditions:

- $(\mathbf{p}_{i+1} - \mathbf{c}_{i+1}) \perp (\mathbf{c}_{i+1} - \mathbf{c}_i)$
- $\det(\mathbf{p}_i - \mathbf{c}_i\,, \mathbf{v}(\mathbf{p}_i)\,, \mathbf{p}_{i+1} - \mathbf{c}_{i+1}) = 0$, i.e., $(\mathbf{p}_{i+1} - \mathbf{c}_{i+1})$ lies in the plane given by the 3 points $\mathbf{c}_i, \mathbf{p}_i, \mathbf{p}_i + \mathbf{v}(\mathbf{p}_i)$.
- $\|\mathbf{p}_{i+1} - \mathbf{c}_{i+1}\| = \|\mathbf{p}_i - \mathbf{c}_i\|$

Figure 4.1a gives an illustration. In a similar way we compute the flow ribbon which belongs to the separation surface of $\mathbf{x}_R$.

Figure 4.2 shows a saddle connector between two focus saddles. This is a part of a flow behind a cylinder. The data set will be explained in section 4.4, where we apply all the above described extraction techniques.

4.2 Features induced by the Boundary



Boundary switch curves partition the boundary of a 3D vector field into regions of inflow and outflow. Two separation surfaces start from there, one in forward and one in backward direction. They may intersect each other or intersect with the separation surfaces emanating from saddles. These intersections are certain stream lines called boundary switch connectors. In the following we deal with the extraction of all these features. Separation surfaces emanating from the boundary as well as boundary switch connectors can be extracted using the Unified Feature Extraction Architecture. However, it is not applicable to the extraction of boundary switch curves. We concentrate on the 3D case and refer to [dLvL99a] for the 2D case.

We start with the extraction of boundary switch curves. Assuming $\mathbf{v}$ to be piecewise trilinear and $z = z_{min}$ being a grid plane of the underlying grid, the boundary switch curves are the zeros of the piecewise biquadratic scalar field $w(x,y,z_{min})$ (where w denotes the third component of $\mathbf{v}$). It is a well-known fact that these isocurves consist of piecewise hyperbolas with G^0 continuous junction points. We extract and describe each hyperbolic curve segment as a rational quadratic Bézier curve $\mathbf{h}(t)$ described by the Bézier points $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ and the corresponding weights g_0, g_1, g_2 with $g_0 = g_2 = 1$. This way, $\mathbf{b}_0$ and $\mathbf{b}_2$ are the intersections of the curve with the grid lines (obtained by a linear interpolation), $\mathbf{b}_1$ is obtained as the intersection of the tangents at $\mathbf{b}_0$ and $\mathbf{b}_2$ (which are computed as the gradients of $w(x,y,z_{min})$), and g_1 is chosen such that $w(\mathbf{h}(t)) \equiv 0$. Figure 4.3 gives an illustration.

As discussed in section 2.2.2 we need to distinguish between inbound and outbound segments on a boundary switch curve since the separation surfaces emanate from inbound segments only. These segments are separated by inout points. If $\mathbf{v}$ is a piecewise

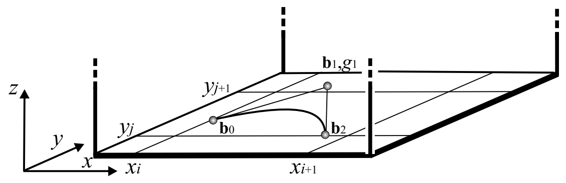
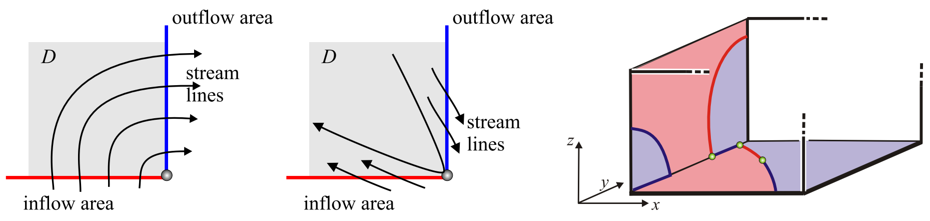


Figure 4.3: Description of a hyperbolic segment as a piecewise rational quadratic Bézier curve.



(a) Boundary switch point on a vertex of the rectangular domain of a 2D vector field.
(b) Inbound segments on the edges of D do not exist.
(c) Inflow/outflow behavior on two faces of D creates two boundary switch curves on the shared edge, which are always outbound segments.

Figure 4.4: Boundary switch curves on the edges of D.

trilinear vector field, all junction points of the piecewise hyperbolic segments are candidates for being an inout point. In addition, up to 4 inout points may exist inside a hyperbolic boundary switch curve segment. Therefore we have applied numerical methods to get the inout points and thus find the inbound and outbound segments.

If $\mathbf{v}$ is piecewise linear, then the boundary switch curves are piecewise linear curves, and inout points can be the junctions of the linear curve segments as well as up to one additional point inside each linear segment. This point can directly be computed by solving a 2×2 system of linear equations.

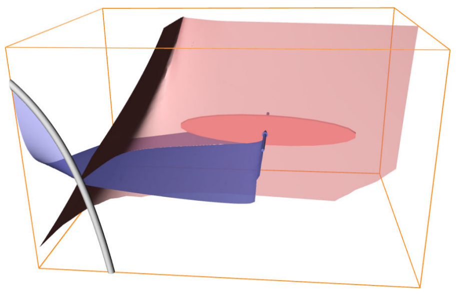
Up to now we only considered boundary switch curves on the faces of the domain D (2.23). However, the edges of D are candidates for being boundary switch curves as well, since the two faces sharing an edge may have a different inflow/outflow behavior. (Figure 4.4a gives a 2D example of a boundary switch point on a vertex of a rectangular domain.) Hence, also the edges of D have to be checked for being boundary switch curves. This is done by computing the inflow/outflow behavior of the faces of D which share the edge. Figure 4.4c illustrates an example.

Boundary switch curves on edges are always outbound segments which therefore do not create separation surfaces. To show this, consider figure 4.4b. If a point on the edge of D is an inbound point, it must have a curvature diverging to infinity. Since this only happens for critical points [WT02] and critical points are not supposed to be on the boundary, an inbound segment on an edge cannot exist.

Starting from the inbound segments, we compute two separation surfaces – one in forward and one in backward direction – by applying the algorithm for integrating stream surfaces. These are seeded by placing a number of starting points on an inbound segment in an equidistant manner.

Similar to the previously described saddle connectors, we can find boundary switch connectors by applying the algorithm for intersecting stream surfaces to the sets of all attracting and all repelling separation surfaces in the flow, i.e., emanating from the boundary or saddles. Boundary switch connectors are all intersections involving a separation surface from the boundary. Figure 4.5a gives an example.

In order to depict boundary switch connectors we choose a double flow ribbon rep-


(a) Intersection of separation surfaces.

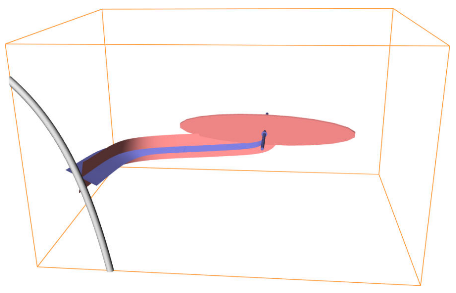

(b) Boundary switch connector depicted as double flow ribbon.

Figure 4.5: Boundary switch connector between a saddle and a boundary switch curve.

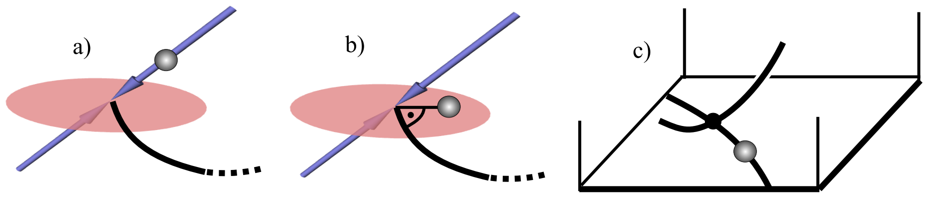
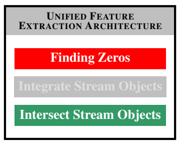


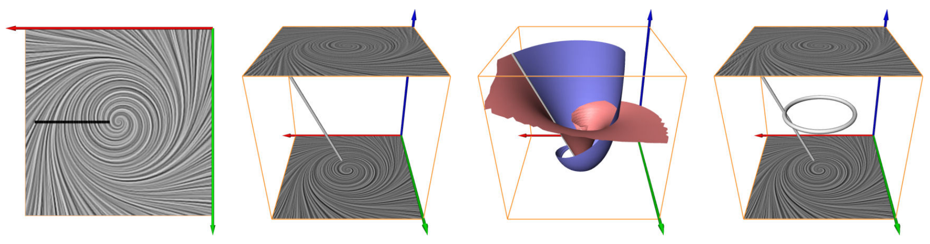
Figure 4.6: Starting points (gray balls) for the double flow ribbon integration; a) for saddle connectors; b) for boundary switch connectors starting in a saddle; c) for boundary switch connectors starting on a boundary switch curve.

resentation similar to the one described for saddle connectors. However, we need a modification of this, since the original approach made use of the inflow/outflow direction of the saddles (Figure 4.6a) and there is no appropriate equivalent for boundary switch curves. Here, we have to choose the starting point in a different way: if the separation surface starts in a saddle, we place the starting point into the inflow/outflow plane, perpendicular to the direction of the boundary switch connector, with a certain predefined distance ε to the saddle point (Figure 4.6b). If the boundary switch connector starts from a boundary switch curve, the starting point is located on the boundary switch curve in a certain distance ε to the starting point of the connector (Figure 4.6c). Figure 4.5b shows the resulting visual representation.

4.3 Closed Stream Lines

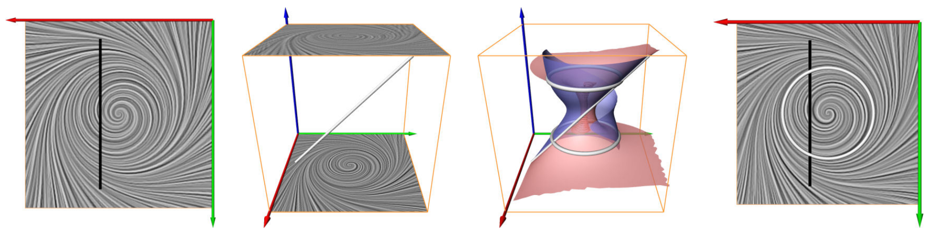


Isolated closed stream lines may appear in flows with non-zero divergence. They are global topological features and are important since they have a separation property in 2D flows: they separate the vector field into two areas of different flow behavior. Wischgoll et al. presented in [WS01] an algorithm to extract closed stream lines in vector fields defined on triangular grids by checking whether a stream line re-enters a triangle. In contrast to this, the algorithm presented here is independent of the underlying grid. We make use of the stream surface intersection algorithm for finding a single closed stream line. In order to find all of them we take the locations of sources and sinks into account. The considerations are restricted to 2D vector fields.



(a) **v** and line ℓ. (b) **w** and **c**. (c) Stream surfaces started from **c**. (d) Detected intersection curve is closed stream line.

Figure 4.7: Detecting closed stream lines crossing a line ℓ.



(a) **v** and line ℓ. (b) **w** and **c**. (c) Closed stream line detected twice. (d) Projection onto $z = 0$. One closed stream line remains.

Figure 4.8: Detection and projection of a closed stream line crossing a line ℓ twice.

Given a 2D vector field **v** and a line

$$\ell(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \tag{4.1}$$

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

$$\mathbf{w}(x,y,z) = \begin{pmatrix} u(x,y) \\ v(x,y) \\ 0 \end{pmatrix} \;, \; \mathbf{c}(t) = \begin{pmatrix} x(t) \\ y(t) \\ t \end{pmatrix} . \tag{4.2}$$

Note that **c** is strictly monotonous in z-direction: for each z-value there is only one point on **c**. Then we apply the stream surface intersection described in section 3.3.3 in **w** by seeding from $\mathbf{c}(t)$ in both forward and backward direction. This way, each detected intersection curve in **w** corresponds to a closed stream line in **v**. Since a line in **w** always consists of points with the same z-value, simply an omitting of the z-values of the found intersection curves in **w** yields the closed stream lines in **v**. Figure 4.7 illustrates this algorithm.

Note that the algorithm detects the same closed stream line more than one time if it crosses ℓ more than one time. Figure 4.8 shows an example. The removal of multiple detected closed stream lines in **w** can be done by projecting them onto the plane $z = 0$ and comparing their Hausdorff distance: if it is under a certain small threshold, one of the closed curves is deleted.

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

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

This theorem follows directly from the index theorem of 2D vector fields [Asi93]: considering the area of $\mathbf{v}$ inside a closed stream line, this area has a global index of $+1$. Hence, at least one critical point with the index $+1$ must be contained within the area inside the closed stream line.

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

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

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

1. Detect all critical points $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n$ of $\mathbf{v}$ with an index $+1$.
2. Detect the point $\mathbf{p}_0$ on the boundary of the domain of $\mathbf{v}$ which is closest to $\mathbf{p}_1$.
3. The polygon $(\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_n)$ is the line ℓ (Figure 4.9a).

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

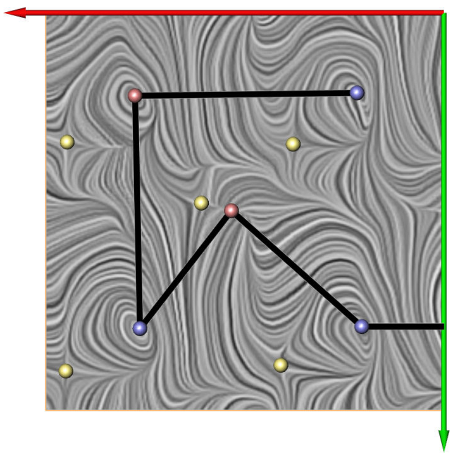
As the integration at and near the critical points is numerically unstable, these small areas must be excluded (Figures 4.9a-b). To achieve this, we split the polygon ℓ into its line segments $(\mathbf{p}_i, \mathbf{p}_{i+1})$. Every $\mathbf{p}$ coinciding with a critical point is moved away from it by a small amount ε along the line segment vector $\mathbf{s}_i = \mathbf{p}_{i+1} - \mathbf{p}_i$. This is done for every segment separately. Thus, we yield the new segments:

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

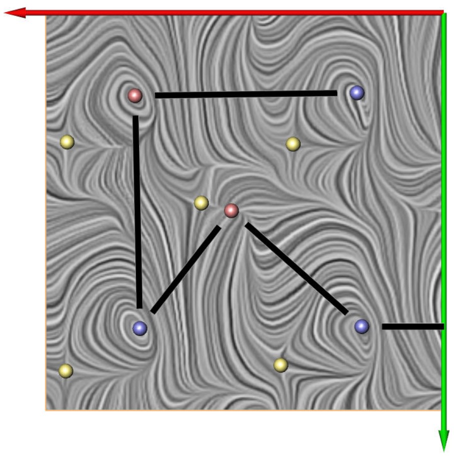
There are two ways to transform the line segments into 3D seeding curves:

1. Transform each point according to its location t on ℓ. This yields seeding curves with non-overlapping z-value ranges. Thus, the algorithm can be applied to all curves at once (Figure 4.9c).
2. Transform each point according to its location t on the line segment itself. The resulting seeding curves have overlapping z-value ranges. Therefore, each curve must be treated separately (Figure 4.9d).

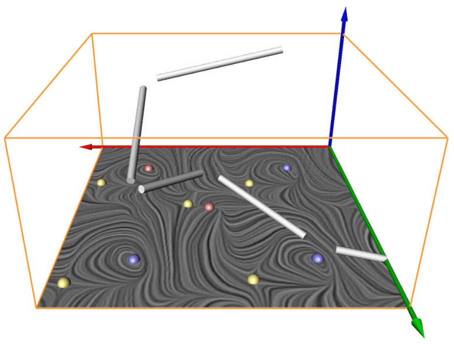
We found the latter type of transformation to be more robust, as the seedings curves have a fixed z-value range and a guaranteed minimal length. In this case, choosing a fixed resolution for the stream surface integration of all curves yields to robust results for our test data sets. Though a different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



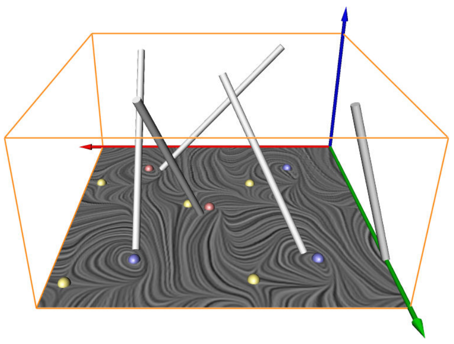
(a) Polygon ℓ connecting all sources/sinks and one boundary point.



(b) Polygon ℓ splitted around the critical points.



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



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

Figure 4.9: Finding seeding lines to detect all closed stream lines of $\mathbf{v}$.

4.4 Applications

In this section we exemplify the extraction techniques described in this chapter by applying them to parameter-independent vector fields. First, we extract the topological skeleton – including saddle and boundary switch connectors – of two 3D vector fields of a rather complex topology. One of these data sets is a flow field *(Cylinder)*, while the other one is a gradient field *(Benzene)*. Table 4.1 gives a summary of these data sets and shows the topological richness of both. Finally, we apply the technique for extracting closed stream lines to a random 2D data set.

Figures 4.2, 4.10, 4.11 and 4.12 visualize a snapshot of a transitional wake behind a circular cylinder [ZFN$^+$95]. This flow exhibits periodic vortex shedding leading to the well known von Kármán vortex street. This phenomenon plays an important role in many industrial applications, like mixing in heat exchangers or mass flow measurements with vortex counters. However, this vortex shedding can lead to undesirable periodic forces on obstacles, like chimneys, buildings, bridges and submarine towers.

This data set was derived from a direct numerical simulation of the Navier-Stokes equation by Gerd Mutschke (FZ Rossendorf). The data resolves the so-called ‘mode

detected features	Benzene	Cylinder
critical points (attracting / repelling saddles)	184 (78 / 43)	13 (7 / 6)
boundary switch curves / inbound segments	12 / 16	13 / 22
attracting / repelling separation surfaces	94 / 59	29 / 28
connectors (overall)	181	59
saddle connectors	129	9
boundary switch connectors **B** $\leftrightarrow$ **B** / **B** $\leftrightarrow$ **Sa**	24 / 28	29 / 21

Table 4.1: Number of features detected by our algorithms.

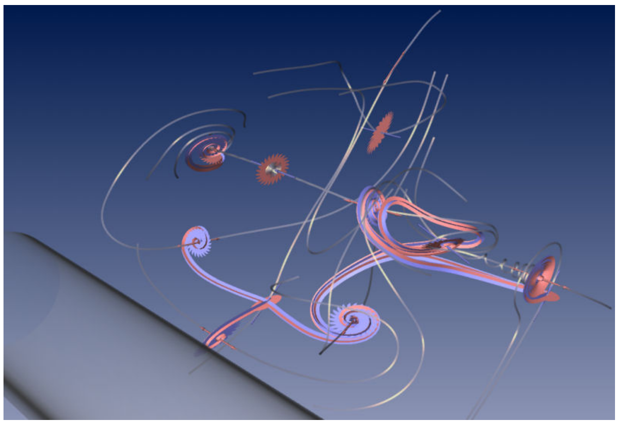


Figure 4.10: Flow behind a circular cylinder. Topological skeleton using saddle connectors.

A' of the 3D transition at a Reynolds number of 200 and at a spanwise wavelength of 4 diameters. The figures display a small near-wake region of a large computational domain. All 13 critical points are contained in the shown domain and on its boundaries 13 boundary switch curves (with 22 inbound segments) are observed. Together they span the topological skeleton of the incompressible velocity field.

The topology enables to reduce a high-dimensional data set to a simple conceptual flow representation from which qualitative conclusions can be drawn. Using saddle connectors as shown in figures 4.10 and 4.11, the skeleton elucidates the symmetry of the mode A with respect to a plane which is perpendicular to the cylinder axis. The spanwise and transverse connectedness of the distributed saddle points of a single snapshot already indicates the experimentally observed good mixing properties of vortex shedding. The saddle points, for instance, are regions of enhanced stretching. Our algorithm detects and visualizes 9 saddle connectors and 50 boundary switch connectors in the data set.

Figure 4.12a shows all separations surfaces emanating from the boundary switch curves and the saddles. The inspection of this figure suggests a high amount of circulating flow behavior in the data set, but due to the occlusion effects introduced by the separation surfaces neither the flow behavior on the boundaries nor the critical points can be seen easily. This complicates further examinations to a high degree. On the other hand, the simplified topological skeletons using saddle and boundary switch connectors give less cluttered visualizations.

Figures 4.13 – 4.15 visualize the electrostatic field around a benzene molecule. This data set was calculated on a 101^3 regular grid using the fractional charges method described in [SS96]. It consists of 184 first order critical points. The separation surfaces shown in figure 4.14b emanate from 78 attracting and 43 repelling saddles. They

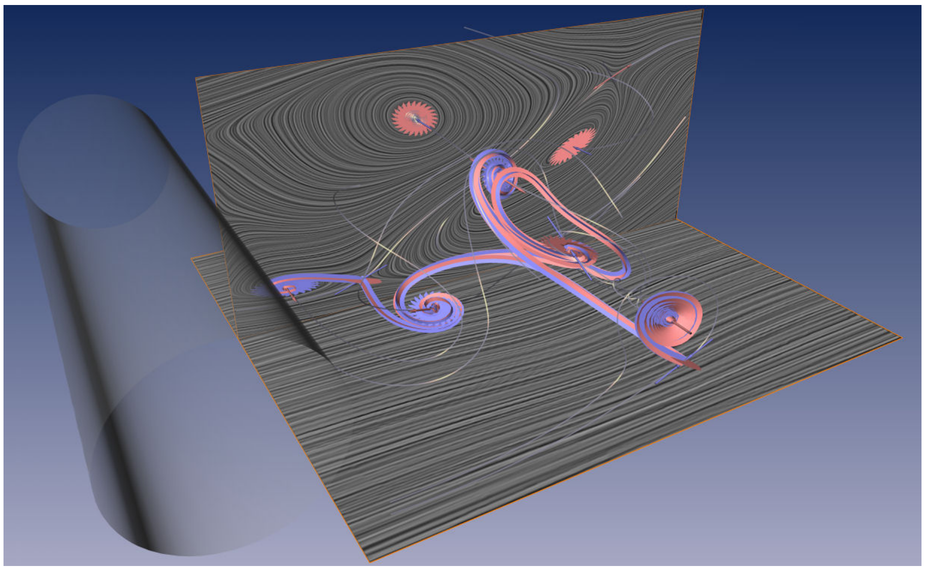
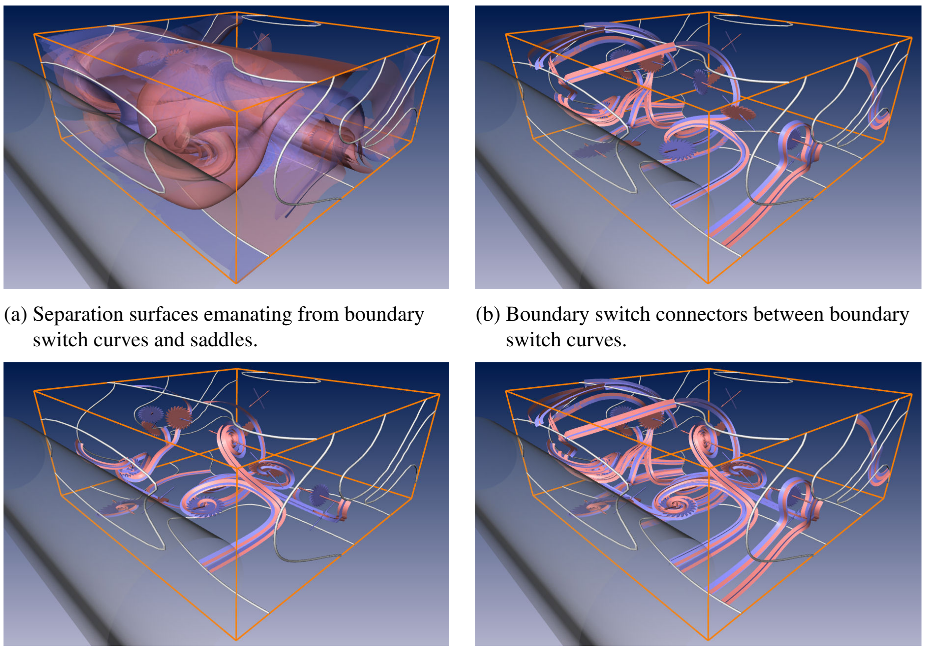


Figure 4.11: Flow behind a circular cylinder. 13 critical points and 9 saddle connectors have been detected and visualized. Additional LIC planes have been placed to show the correspondence between the skeleton and the flow.



(a) Separation surfaces emanating from boundary switch curves and saddles.

(b) Boundary switch connectors between boundary switch curves.

(c) Boundary switch connectors between saddle points and boundary switch curves.

(d) Saddle connectors and both types of boundary switch connectors.

Figure 4.12: Flow behind a circular cylinder. Different topological representations.

hide each other as well as the critical points. Even rendering the surfaces in a semi-transparent style does not reduce the visual clutter to an acceptable degree. Figure 4.14c shows a possible solution to this problem by showing the 129 saddle connectors that we found in this data set. Despite the fact that saddle connectors can only indicate the approximate run of the separation surfaces, the resulting visualization gives more insight into the symmetry and three-dimensionality of the data set. Saddle connectors are a useful compromise between the amount of coded information and the expressiveness of the visualization for complex topological skeletons.

In figure 4.15 we examined the boundary of the benzene data set. For this visualization we used a smaller subdomain around the molecule itself. All 184 critical points are inside the shown region. Shown are the boundary switch curves, the resulting separation surfaces as well as the boundary switch and saddle connectors.

The cylinder and the benzene data set show that boundary switch and saddle connectors give expressive visualizations even for topologically complex 3D vector fields.

Figures 4.9 and 4.16 visualize a random 2D data set defined on a 20×16 grid. Random vector fields are useful tools for a proof-of-concept of topological methods, since they contain a maximal amount of topological information.

We detected 5 saddle points (yellow), 3 sinks (blue) and 2 sources (red) in this data set. Sources and sinks have an index of $+1$. How the polygon ℓ and the corresponding seeding curves are constructed from this information can be seen in figure 4.9. Note, that here the excluded area around the critical points has been enlarged for demonstration purposes only. For the calculations presented in figure 4.16 the excluded area is an order of magnitude smaller. This difference can be seen by comparing figures 4.9d and 4.16d.

Figures 4.16a-b show the stream surfaces emanating from the seeding curves. As discussed in section 4.3 only the two surfaces coming from the same seeding curve are tested for intersection against each other.

Figure 4.16d shows in conjunction with figure 4.16c that our algorithm found a stream surface intersection at all places where a seeding curve crosses a closed stream line. Thus, some of the closed stream lines are detected multiple times. As explained in section 4.3 this can be dealt with by projecting them onto the plane $z = 0$ and comparing their Hausdorff distance. This reduces the number of 9 found intersections to 5 unique closed stream lines – as it can be seen in figure 4.16c. Our algorithm needed approximately 90 seconds to extract the closed stream lines (Pentium 4, 1.7GHz).

One of the closed stream lines in the upper left corner of figure 4.16c is very close to a critical point (a source). Figure 4.16e shows a closeup of that area. This exemplifies that our algorithm works reliable even in a rather small distance away from a critical point.

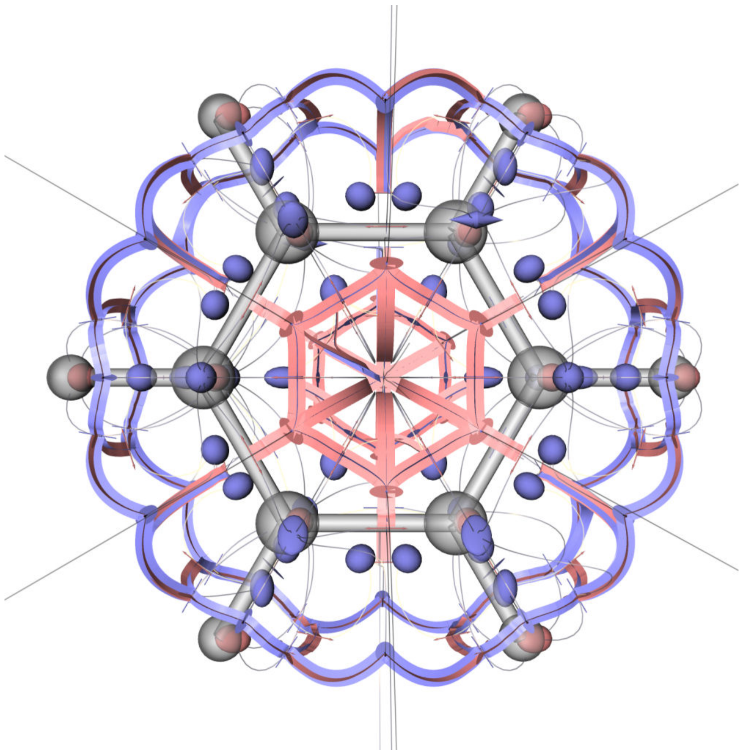
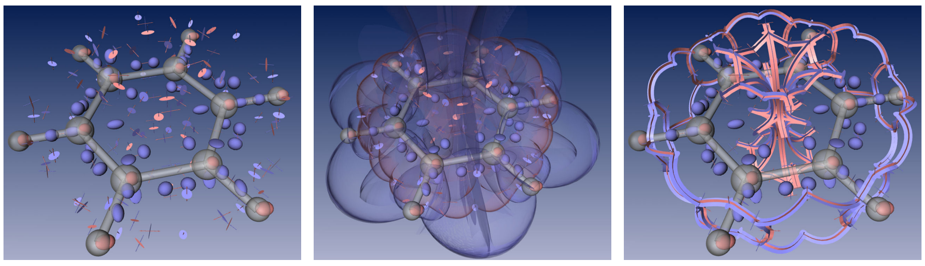


Figure 4.13: Benzene data set: critical points, separation curves and saddle connectors.



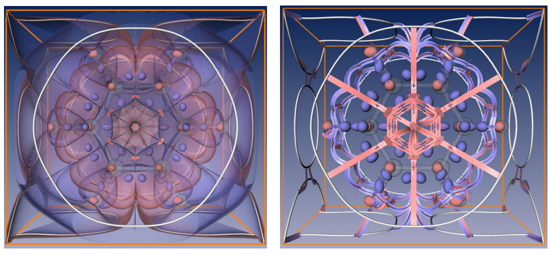
(a) Iconic representation.

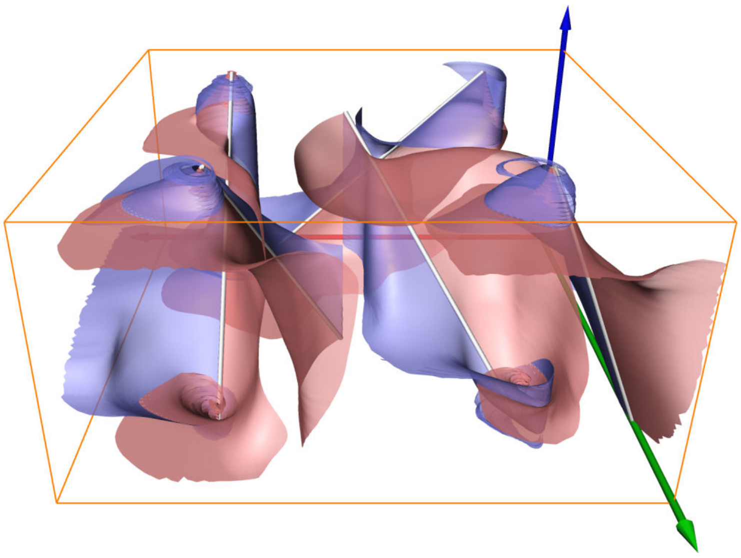
(b) Due to the shown separation surfaces, the topological skeleton of the vector field looks visually cluttered.

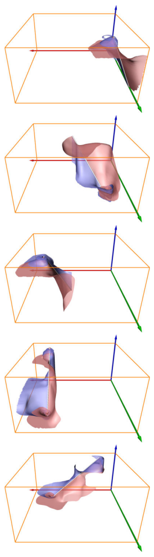
(c) Visualization of the topological skeleton using saddle connectors.

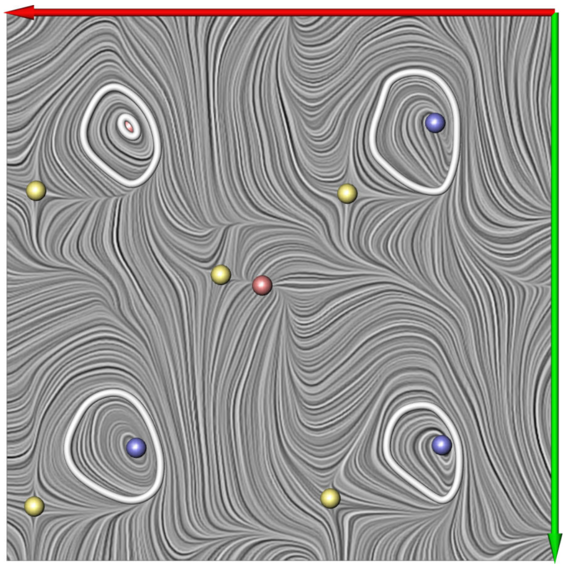
Figure 4.14: Topological representations of the benzene data set with 184 critical points.

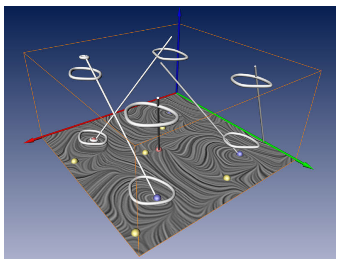
Figure 4.15: Benzene data set with boundary switch curves. (left) Visual clutter due to the shown separation surfaces. (right) Visualization using connectors.




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


(b) Each curve treated separately.


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


(d) 9 detected intersection curves.

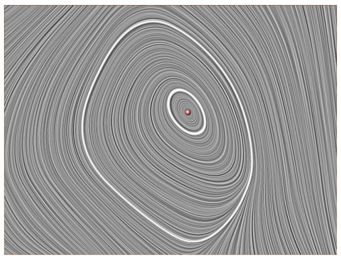

(e) Closeup of the upper left corner.

Figure 4.16: Test data set: 5 closed stream lines have been detected.

Chapter 5

Extraction of Parameter-Dependent Topological Structures

In this chapter we propose algorithms for capturing the topology of parameter-dependent vector fields. We do this for vector fields depending on one (section 5.1) and two parameters (section 5.3). We put a strong focus on extracting all occurrences of all critical points in such data sets as well as the local bifurcations related to critical points. Since parameter-dependent data sets tend to be very large and sometimes even larger than main memory, we designed our methods to be efficient with out-of-core data handling (section 5.1.1). Furthermore, we give algorithms for extracting global bifurcations and tracking closed stream lines (section 5.1.2). In sections 5.2 and 5.4 we apply our techniques to one- and two-parameter-dependent vector fields respectively.

The work presented in this chapter extends the original work on tracking critical points using Feature Flow Fields [TS03]. However, apart from this foundation all techniques presented in this chapter have been newly developed in the course of this thesis. This includes guarantees for finding all occurrences of all critical points, the out-of-core strategy, the handling of global features as well as the complete work on two-parameter-dependent vector fields. Note that all given algorithms are based on the Unified Feature Extraction Architecture.

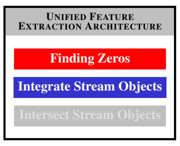
5.1 One-Parameter-Dependent Topology

One-parameter-dependent vector fields are of the form $\mathbf{v}(\mathbf{x},t)$ and we are interested in how the topological structures move or change otherwise with changing parameter t. As an example, critical points move with changing t and form line-type structures now. Moving topological structures might enter or leave the domain as well as appear or disappear inside the domain at certain well-defined bifurcations. It is the goal to not only extract the moving topological structures, but also the bifurcations which mark important structural changes of the topological skeleton.

The parameter t can have different interpretations – a very common one is physical time. In order to ease the explanations we will speak mostly of time-dependent vector fields in the following.

We first show how to find all occurrences of all critical points in time-dependent data sets (section 5.1.1) before we introduce new methods to detect certain global bifurcations like saddle connections and cyclic fold bifurcations (section 5.1.2).

5.1.1 Tracking Critical Points



Critical points are important topological features and tracking their locations over time is necessary for capturing the dynamics of a flow. This is equivalent to extracting the zero lines of **s** (as given by (2.14)), where all points on these lines are zero points of **v** at a certain time. To do so, one can extract and connect the zeros on the faces of an underlying piecewise linear grid as proposed by Tricoche et al. [TSH01b] for 2D time-dependent vector fields and by Garth et al. [GTS04b] for the 3D case. Since this works only for triangular or tetrahedral grids, we follow a different approach: based on the Unified Feature Extraction Architecture we choose a FFF-based technique for extracting the paths of the critical points of **v** as stream lines of a certain derived feature flow field **f**. This approach has been presented first by Theisel and Seidel in [TS03]. In the course of this thesis we refined this by giving a set of seeding points for the integration which guarantees to find all occurrences of critical points and is compatible to an out-of-core data handling following the ideas of section 3.1.1. Besides tracking the locations of the critical points we are also interested in corresponding structural changes like Hopf bifurcations. As we will show in the following, locations and properties of critical points in one-parameter-dependent vector fields can be assessed using the algorithms for finding zeros and integrating stream lines only.

Under the assumption that we already know the position of a critical point at a certain time t_0, we are interested in tracking its location over time – not necessarily only forward, but also backwards in time. For this we construct a vector field **f** in the space-time domain with the following properties: for any two points $\mathbf{x}_0$ and $\mathbf{x}_1$ on a stream line of **f**, it holds $\mathbf{v}(\mathbf{x}_0) = \mathbf{v}(\mathbf{x}_1)$. This means that a stream line of **f** connects locations with the same values of **v**. Figure 5.1 gives an illustration. In particular, if $\mathbf{x}_0$ is a critical point in **v**, then the stream line of **f** describes the path of the critical point over time.

In other words, the feature flow field in question describes the direction in space-time in which all components of **v** locally remain constant. For a 2D time-dependent vector field this is the direction perpendicular to the gradients of the two components of **v**. We get

$$\mathbf{f}(x,y,t) = \mathrm{grad}(u) \times \mathrm{grad}(v) = \begin{pmatrix} u_x \\ u_y \\ u_t \end{pmatrix} \times \begin{pmatrix} v_x \\ v_y \\ v_t \end{pmatrix} = \begin{pmatrix} \det(\mathbf{v}_y, \mathbf{v}_t) \\ \det(\mathbf{v}_t, \mathbf{v}_x) \\ \det(\mathbf{v}_x, \mathbf{v}_y) \end{pmatrix}. \qquad (5.1)$$

The FFF for 3D time-dependent vector fields is a straightforward extension of the 2D case. The 4D FFF **f** is defined by the conditions

$$\mathbf{f} \perp \mathrm{grad}(u) = (u_x, u_y, u_z, u_t)^T \ , \ \mathbf{f} \perp \mathrm{grad}(v) \ , \ \mathbf{f} \perp \mathrm{grad}(w). \qquad (5.2)$$

This gives a unique solution for **f** (except for scaling)

$$\mathbf{f}(x,y,z,t) = \begin{pmatrix} +\det(\mathbf{v}_y, \mathbf{v}_z, \mathbf{v}_t) \\ -\det(\mathbf{v}_z, \mathbf{v}_t, \mathbf{v}_x) \\ +\det(\mathbf{v}_t, \mathbf{v}_x, \mathbf{v}_y) \\ -\det(\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z) \end{pmatrix}. \qquad (5.3)$$

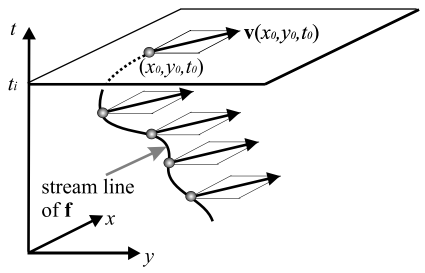


Figure 5.1: Tracking critical points: all points on a stream line of **f** have the same value in **v**. Exemplified using a 2D setup. From [TS03].

The derivation of the feature flow fields for tracking critical points has already been given by Theisel and Seidel in [TS03]. It remains to find a suitable set of starting points which guarantees to find all occurrences of critical points in **v**. Such a set consists of all locations in space-time where a critical point appears for the first time:

- **Intersections with the boundary:** critical points enter (or leave) the domain at these places.
- **Fold bifurcations:** a critical point appears and at the same moment splits up into a saddle and a source/sink/center (or the other way around: a saddle and a source/sink/center collapse and disappear).

As already indicated, critical points may also disappear at the boundary or at fold bifurcations. We have to distinguish between entry and exit events at the boundaries and between birth and death events. In other words, we need a complete set of t-forward points (section 3.1.1). In the following we give the conditions for finding all intersections with the boundaries and all fold bifurcations, and we show how to distinguish between appearance and disappearance. We start with the explanations for 2D time-dependent vector fields.

To find all intersections with the boundaries, we have to solve

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

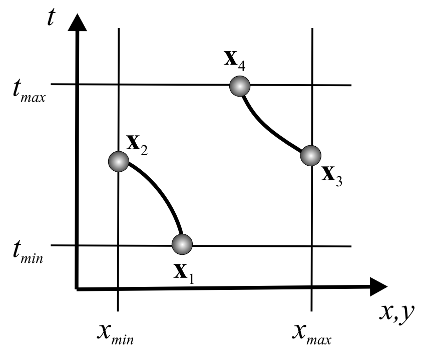
Each of the 6 solutions turns out to be a simple extraction of isolated zeros of a 2D (steady) vector field. We can make the following distinction:

- *Bottom intersection points* are intersections with the plane $t = t_{min}$.
- *Top intersection points* are intersections with the plane $t = t_{max}$.
- *Side intersection points* are intersections with the plane $x = x_{min}$, $x = x_{max}$, $y = y_{min}$, or $y = y_{max}$ respectively.

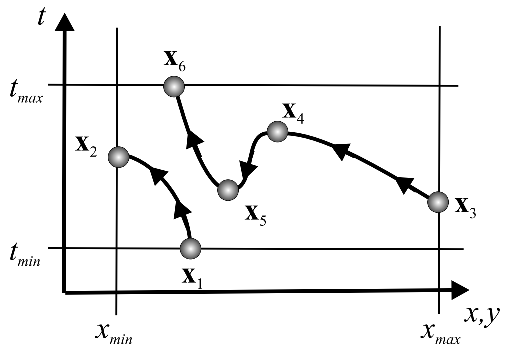
Side intersection points can be further classified into entry and exit points. At an entry point, a t-forward integration of **f** goes into the domain, while at an exit point a t-forward integration leaves it. Figure 5.2a illustrates the different kinds of intersection points with the boundary. It is easy to see that only bottom and entry side intersections are t-forward points.

To detect all fold bifurcations of a 2D time-dependent vector field we search for locations **x** with

$$[\, \mathbf{v}(\mathbf{x}) = (0,0)^T \ , \ \det(\mathbf{J_v}(\mathbf{x})) = 0 \,] \tag{5.4}$$

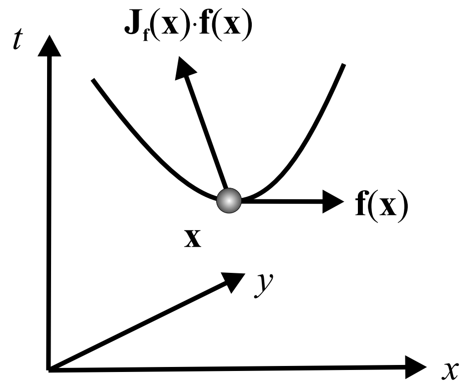


(a) Bottom intersection $\mathbf{x}_1$, exit $\mathbf{x}_2$ and entry $\mathbf{x}_3$ side intersection, top intersection $\mathbf{x}_4$.

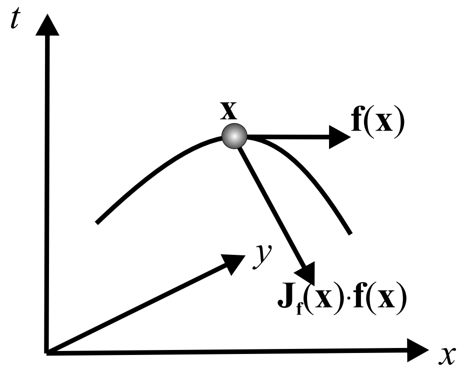


(b) Example consisting of a bottom intersection $\mathbf{x}_1$, exit $\mathbf{x}_2$ and entry $\mathbf{x}_3$ side intersections, death $\mathbf{x}_4$ and birth $\mathbf{x}_5$ fold bifurcations, and a top intersection $\mathbf{x}_6$. The paths of the critical points consist of 4 segments which are integrated t-forward from $\rightarrow$ to: $\mathbf{x}_1 \rightarrow \mathbf{x}_2$, $\mathbf{x}_3 \rightarrow \mathbf{x}_4$, $\mathbf{x}_5 \rightarrow \mathbf{x}_4$, $\mathbf{x}_5 \rightarrow \mathbf{x}_6$.

Figure 5.2: Intersections of the paths of critical points with the domain boundary.



(a) Birth event since the last component is positive.



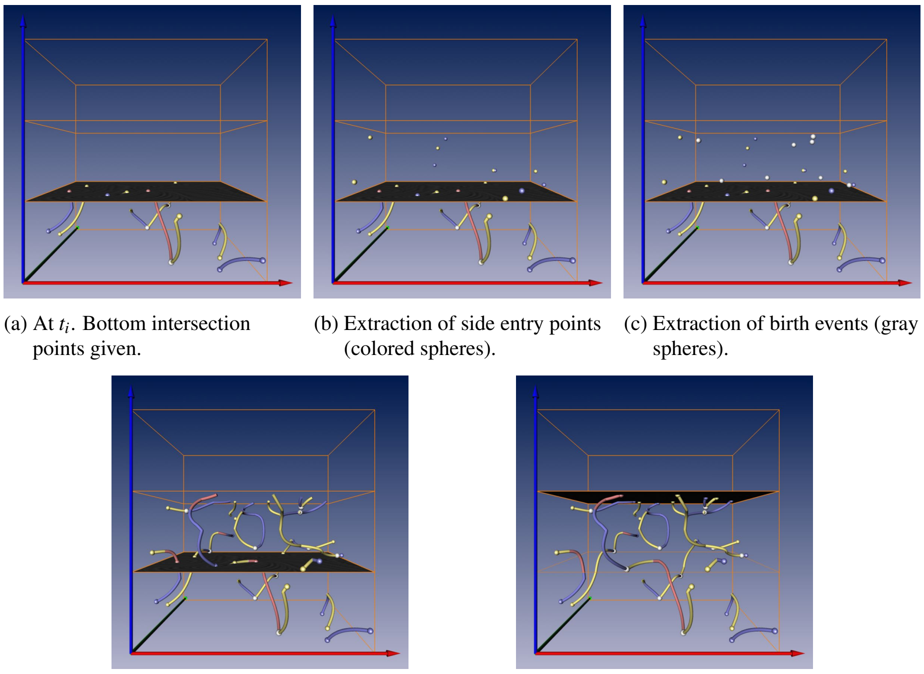
(b) Death event since the last component is negative.

Figure 5.3: Classifying fold bifurcations by the last component of $\mathbf{J_f}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})$.

where $\mathbf{J_v}$ is the Jabcobian matrix of $\mathbf{v}$. Equation (5.1) shows that the second condition of (5.4) ensures that the last component of $\mathbf{f}$ vanishes, i.e., that $\mathbf{f}$ is parallel to the t-axis. To solve (5.4), we simply apply the algorithm for finding zeros. There are two kinds of fold bifurcations: birth and death events. To distinguish them, we consider the last component of $\mathbf{J_f}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})$ at the fold bifurcation. If this component is positive, we have a birth bifurcation. If it is negative, a death bifurcation is present.[1] We call this discrimination of fold bifurcations into births or deaths the *BD classification*. We will extend the BD classification for the use with two-parameter-dependent fields in section 5.3. Figure 5.3 illustrates both types of fold bifurcations. It is easy to see that only birth bifurcations are t-forward points.

For 3D time-dependent vector fields, the extraction of the seeding points follows the same ideas. Boundary intersections are found as isolated critical points of the 3D

[1]If the last component of $\mathbf{J_f}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})$ equals zero, then this gives a structurally unstable event, i.e., an event which disappears by adding noise to $\mathbf{v}$. This is not considered here.


(a) At t_i. Bottom intersection points given.

(b) Extraction of side entry points (colored spheres).

(c) Extraction of birth events (gray spheres).

(d) Integration of $\mathbf{f}$.

(e) At t_{i+1}. Top intersections serve as input to the next interval.

Figure 5.4: Tracking of critical points can be done in one sweep through the data while possibly holding only parts of the data at once.

(steady) vector fields at the space-time domain boundary. Again, only bottom and entry side intersections are of interest. Fold bifurcations are the solutions of

$$[\,\mathbf{v}(\mathbf{x}) = (0,0,0)^T \;\;,\;\; \det(\mathbf{J}_\mathbf{v}(\mathbf{x})) = 0\,] \tag{5.5}$$

which corresponds to numerically finding isolated critical points in 4D vector fields. The distinction between births and deaths follows the 2D case.

We now have a complete set of starting points for integrating $\mathbf{f}$ ((5.1) or (5.3) respectively), which guarantees to find all occurrences of critical points in a time-dependent vector field $\mathbf{v}$. Furthermore, this set consists of t-forward points only and we can apply algorithm 3 from section 3.1.1 in order to track critical points in one sweep: the algorithm reads only parts of the data and each part only once. In contrast to the original approach of [TS03] our algorithm ensures that every path of a critical point is integrated only once. Thus, a removal of multiple integrated stream lines is not needed anymore. Note, that we have to start a *t-forward* integration of $\mathbf{f}$ at the seeding points, which can be a forward or backward integration – see section 3.1.1 for details.

Figure 5.4 illustrates how this algorithm works between two time steps t_i and t_{i+1}. We aim at extracting the paths of all critical points in the time interval between t_i and t_{i+1} and start with the bottom intersection points at t_i (Figure 5.4a). They are either given as the result of the previous time interval, or if $t_i = t_{min}$ they have to be extracted using the algorithm for finding zeros applied to the 2D (steady) vector field $\mathbf{v}(x,y,t_{min})$. Similarly we extract the side entry intersection points (Figure 5.4b), where critical points might enter the domain. In addition to this we need to extract the birth events, i.e., the inner bifurcations where critical points might appear for the first time (Figure 5.4c). This is done by finding zeros in the 3D vector field given by (5.4). Now we integrate stream lines in **f** starting from the bottom, the side and the births (Figure 5.4d). This yields the paths of all critical points in the considered time interval and the top intersections are the bottom intersections of the next one, i.e., they serve as input for the algorithm in the next time interval (Figure 5.4e).

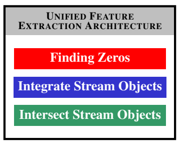
Note that t_i and t_{i+1} do not necessarily have to be two consecutive time steps given by the underlying time-dependent data set. Any number of physically given time steps can be between them. In fact, t_i and t_{i+1} can be arbitrarily chosen as long as $t_i < t_{i+1}$. If they equal t_{min} and t_{max}, then the whole data set will be treated at once. Any subdivision of the temporal axis yields the same result. Hence, it can entirely be subdivided according to memory requirements.

After tracking the critical points we have to determine their type, i.e., distinguish between saddles, sinks, sources, and centers. In our implementation, we do this by applying the eigenanalysis known from the steady case (section 4.1) to each point of the path. However, excluding certain degenerated cases (where the Jacobian matrix **J** has not full rank) the problem reduces to this: given that we are integrating in t-forward direction only, the type of a critical point does not change along its path with the exception of a Hopf bifurcation, which denotes the location where a sink becomes a source or vice versa. Thus, this denotes the location of a center, i.e., a critical point with a vanishing divergence and a positive Jacobian. Hopf bifurcations can be extracted similar to fold bifurcations by numerically solving the system

$$[\mathbf{v} = \mathbf{0},\ \mathrm{div}(\mathbf{v}) = 0] \tag{5.6}$$

and selecting all isolated solutions with positive Jacobian. Again, this is an application of the algorithm for finding zeros. Summarized, we can determine the type of a critical point at its entry or birth event and keep this type until its exit or death while possibly switching between sinks and sources at Hopf bifurcations.

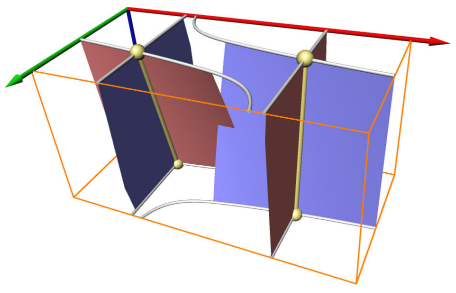
5.1.2 Global Bifurcations



In the previous section we already dealt with local bifurcations of time-dependent vector fields like fold and Hopf bifurcations. Here we present new methods for extracting global bifurcations like saddle connections, periodic blue sky bifurcations, and cyclic fold bifurcations. Furthermore, we show how to track closed stream lines. All these features and events can be extracted using the algorithm for intersecting stream surfaces. Partly, the paths of the critical points and some local bifurcations serve as an input to this. Hence, we need the extraction results of the previous section in order to extract the global bifurcations discussed here. We restrict the discussion to 2D time-dependent vector fields.

Detecting Saddle Connections

Saddle connections are global bifurcations which appear when two separatrices starting from saddle points coincide, i.e., when a separatrix of one saddle ends in another saddle. Figure 2.18 on page 37 gives an illustration. We are not aware of pre-existing solutions to extracting all saddle connections of **v** for visualization purposes.


(a) Separation surfaces starting from the paths of the saddle points.

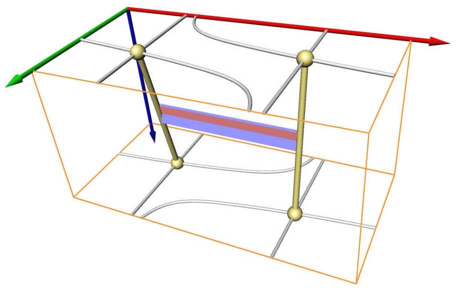

(b) Saddle connection as the intersection of these surfaces.

Figure 5.5: Extracting saddle connections.

The solution we propose here uses the algorithm for intersecting stream surfaces. We start these stream surfaces at the critical lines of **s** which represent a moving saddle. In fact, we start four stream surface integrations[2] at the critical lines of **s** into the directions of the eigenvectors of the Jacobian matrix. This gives the separation lines of the 2D saddles swept over time, i.e., surfaces in the 3D space-time domain. Their intersection yields a stream line of **s** with the property of being a separatrix of both saddles – starting at one saddle into its outflow direction and ending in the other saddle into its inflow direction, i.e., a saddle connection. Figure 5.5 gives an illustration.

A special case of saddle connections is the so-called *periodic blue sky bifurcation* ([AS92]) where two separatrices of the same saddle coincide. Our algorithm to extract saddle connections automatically extracts these bifurcations as well. Figure 5.6 illustrates this.

Tracking Closed Streamlines

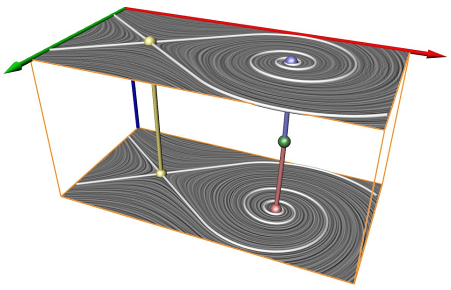
Closed stream lines evolve over time in **v** and several bifurcations can occur: a closed stream line may enter or exit the domain, or two closed stream lines may collapse and disappear (or the other way around). The latter case is called *cyclic fold bifurcation* and is illustrated in figure 2.20 on page 37.

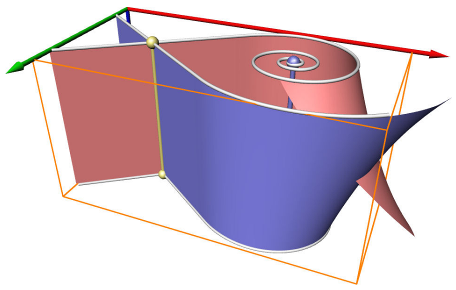
A first approach to track closed stream lines was proposed in [TSH01b]: closed stream lines are extracted in different time levels, and corresponding stream lines in adjacent time levels are connected. The results are tube-shaped surfaces starting/ending in Hopf bifurcations, periodic blue sky bifurcations, or at the boundaries of the domain. This approach depends on the underlying grid structure and does not consider cyclic fold bifurcations.

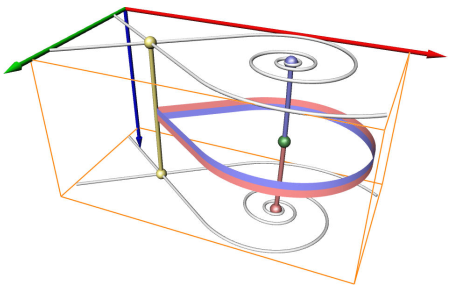
We present a new solution for tracking closed stream lines without these restrictions. First we describe the primary part of the algorithm by assuming a starting closed stream line is already given. How to obtain such a starting line is explained afterwards.

Our new approach follows the general idea of feature flow fields (section 3.1). Suppose we already have a closed stream line $\mathbf{c}_i$. We would like to construct a 3D vector field **g** such that the evolution of $\mathbf{c}_i$ over time can simply be obtained by a stream surface integration of **g** starting at $\mathbf{c}_i$. Unfortunately, such a feature flow field **g** cannot locally be derived from **s**, since the evolution of a closed stream line is a global process. But the basic principle of feature flow fields can be reduced to this: given $\mathbf{c}_i$, we want to find

[2]Two forwards and two backwards.


(a) Critical lines of **s** and two LIC planes.


(b) Separation surfaces shortly after their intersection.


(c) Two separation curves of the same saddle coincide; the intersection with the yellow line is the location of the saddle creating the bifurcation.

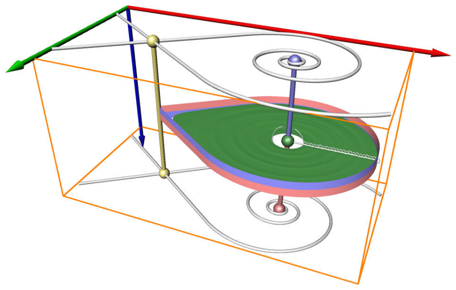
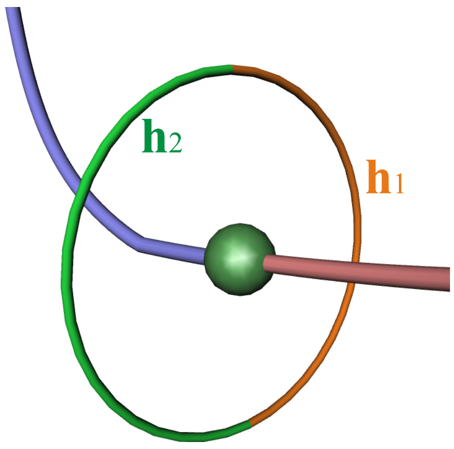

(d) Tracked closed stream line starting from Hopf bifurcation.

Figure 5.6: Periodic blue sky bifurcation.

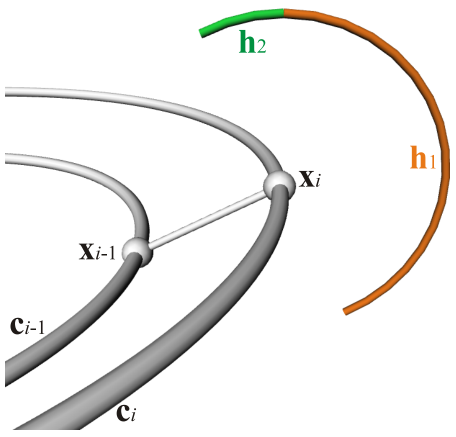
an adjacent closed stream line $\mathbf{c}_{i+1}$ within a certain distance d from $\mathbf{c}_i$. Furthermore, d shall refer to the (x,y,t)-domain, i.e., we do not make any assumption whether the t-value of $\mathbf{c}_{i+1}$ is before, after, or at the same t-value as $\mathbf{c}_i$. Note, that a (closed) stream line is uniquely defined by a single point on it. Hence, we only have to construct a point $\mathbf{x}_{i+1}$ on $\mathbf{c}_{i+1}$ with a certain distance from a point $\mathbf{x}_i$ on $\mathbf{c}_i$. To achieve this, we apply the concept of connectors.

We describe one step of our algorithm now (Figures 5.7b-d). Given two adjacent closed stream lines $\mathbf{c}_{i-1}$ and $\mathbf{c}_i$ together with their defining points $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$ (Figure 5.7b), we want to find a point $\mathbf{x}_{i+1}$ which defines the next closed stream line $\mathbf{c}_{i+1}$. To do so, we consider a plane through $\mathbf{x}_i$ perpendicular to $\mathbf{s}(\mathbf{x}_i)$ and place a circle $\mathbf{k}$ around $\mathbf{x}_i$ with the radius d into this plane: every point on $\mathbf{k}$ represents a certain step in space-time. It is easy to see, that if $\mathbf{c}_{i+1}$ is actually existing, then $\mathbf{x}_{i+1} \in \mathbf{k}$ is fulfilled. Since we assume the closed stream line to evolve continuously, we search $\mathbf{x}_{i+1}$ only on a circular arc $\hat{\mathbf{k}} \subset \mathbf{k}$ consisting of all points $\mathbf{x} \in \mathbf{k}$ with $(\mathbf{x}-\mathbf{x}_i)(\mathbf{x}_i-\mathbf{x}_{i-1}) > 0$ (Figure 5.7b). This ensures, that the algorithm does not run back to $\mathbf{c}_{i-1}$. Anyway, if $\mathbf{c}_{i-1}$ is not given we have $\hat{\mathbf{k}} = \mathbf{k}$. $\hat{\mathbf{k}}$ is further subdivided by cutting along the t-direction into two arcs $\mathbf{h}_1$ and $\mathbf{h}_2$.[3] They act as seeding curves of a stream surface integration of $\mathbf{s}$ in both forward and backward direction (Figure 5.7c). We apply the algorithm for intersecting stream surfaces and yield a connector describing the closed stream line $\mathbf{c}_{i+1}$ (Figure 5.7d).

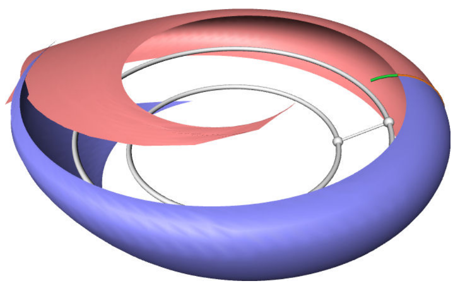
[3]The splitting of $\hat{\mathbf{k}}$ into $\mathbf{h}_1$ and $\mathbf{h}_2$ is necessary to ensure that each found intersection is a closed stream line. If we would start the integration on $\hat{\mathbf{k}}$, the intersection curves could be stream lines starting and ending in different points, i.e., not closed stream lines, since two points on $\hat{\mathbf{k}}$ may have the same t-value.



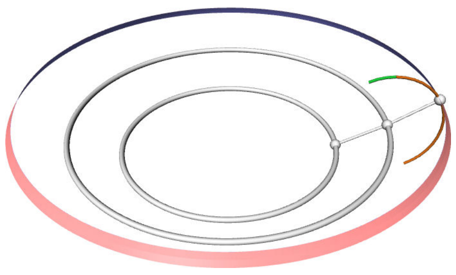
(a) Setup for starting at Hopf bifurcation.



(b) Seeding curves $\mathbf{h}_1$ and $\mathbf{h}_2$ for finding the closed stream line $\mathbf{c}_{i+1}$.



(c) Intersecting stream surfaces evolving from $\mathbf{h}_1$.



(d) Detected connector is $\mathbf{c}_{i+1}$.

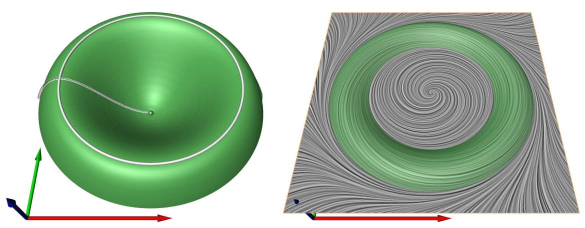
Figure 5.7: Tracking closed stream lines.

Assuming length$(\mathbf{h}_1) \geq$ length$(\mathbf{h}_2)$, we first apply the stream surface integration starting in $\mathbf{h}_1$. This is encouraged by the fact that the direction of the closed stream line's evolution in this step is unlikely to differ much from the previous direction. Our experience shows that this yields much faster execution times. If an intersection curve is found, this is the new closed stream line $\mathbf{c}_{i+1}$; its intersection with $\mathbf{h}_1$ gives $\mathbf{x}_{i+1}$. If no intersection is found, we check $\mathbf{h}_2$ in a similar way. If this gives no result either, the algorithm stops.

After extracting a sequence of closed stream lines, there are two ways for extracting the stream surface describing them. One is to use the curve $[\mathbf{x}_0, \ldots, \mathbf{x}_n]$ as a seeding line for another stream surface integration. However, since we already extracted a number of closed stream lines, we use them to triangulate the strips between each adjacent $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$. The result is a triangular mesh representing a stream surface describing the evolution of a closed stream line in space-time.

Figure 5.8a shows an example of a tracked closed stream line starting at a Hopf bifurcation. We see the sequence of defining points $\mathbf{x}_i$ as well as the searching arcs $\mathbf{h}_1$ and $\mathbf{h}_2$ in every step. This example shows that the algorithm can deal with cyclic fold bifurcations: it appears at the closed stream line $\mathbf{c}_i$ if $(t_i - t_{i-1})(t_{i+1} - t_i) \leq 0$ (where t_i is the t-component of the points on $\mathbf{c}_i$). The example in figure 5.8a has one cyclic fold bifurcation (gray line).

Another example is shown in figure 5.6d, where a Hopf and a periodic blue sky


(a) Stream surface describing closed stream lines. Cyclic fold bifurcation is detected on top of this surface (gray line). The intersection of the surface with a plane $t = \text{const.}$ yields two closed stream lines.

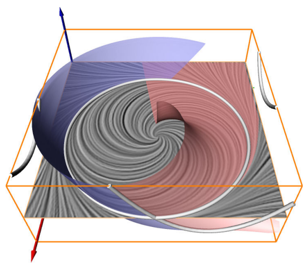

(b) A closed stream line touching the boundary of the domain is obtained by intersecting stream surfaces of **s** starting from the boundary switch curves.

Figure 5.8: Bifurcations of closed stream lines in a 2D time-dependent vector field.

bifurcation have the same t-value. It turns out that the closed stream line completely evolves in the same time slice. This is an example where tracking approaches fail which are based on extracting and connecting closed stream lines in fixed time steps.

The numerical integration of the seeding line $[\mathbf{x}_0, \ldots, \mathbf{x}_n]$ has the advantage that numerical errors are not inherited: if due to numerical errors $\mathbf{x}_i$ is located a little bit away from a closed stream line, $\mathbf{x}_{i+1}$ can still be found correctly.

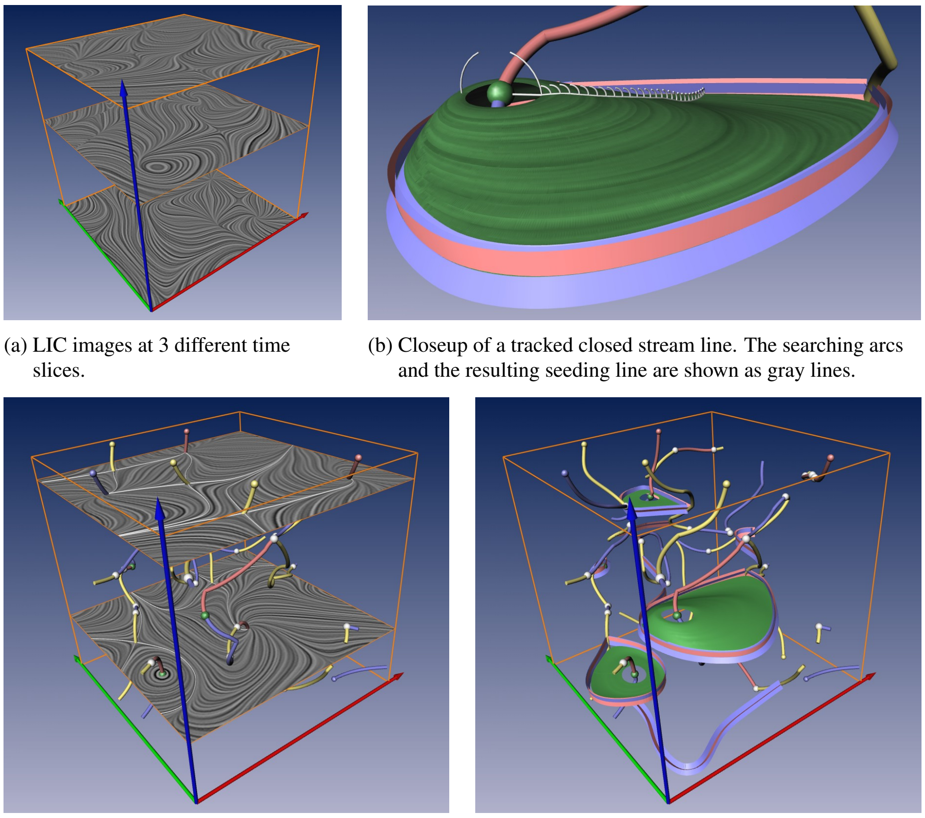
To complete the algorithm of tracking closed stream lines, we still have to find a system of initial closed stream lines, i.e., closed stream lines where the above described tracking starts or ends. This system of initial stream lines has to be chosen in such a way that all closed stream lines are guaranteed to be tracked. To do so, we choose all events where a closed stream line appears or disappears. These events are:

- Hopf bifurcations.
- Periodic blue sky bifurcations.
- Closed stream lines at the first or last time step.
- Closed stream lines touching the boundary of the domain and appearing or disappearing subsequently.
- Cyclic fold bifurcations.

At a Hopf bifurcation $\mathbf{x}_0$, we start the integration in a semi-circle around $\mathbf{x}_0$ which lies in the plane defined by $(0,0,1)^T$ and $(0,0,1)^T \times \mathbf{f}(\mathbf{x}_0)$, where $\mathbf{f}$ is the FFF for tracking critical points as given by (5.1). Figure 5.7a illustrates this. At a periodic blue sky bifurcation, we take any point on the stream line as starting point $\mathbf{x}_0$. Setting $\hat{\mathbf{k}}$ as the full circle $\mathbf{k}$, the rest of the integration step is as described above.

To detect closed stream lines at the first and last time step of $\mathbf{v}$ means to detect all closed stream lines in a 2D steady vector field. This has been treated in section 4.3.

To find closed stream lines touching the boundary of the domain of $\mathbf{v}$, we extract the boundary switch curves (section 4.2) of $\mathbf{s}$. From these lines we start a stream surface integration of $\mathbf{s}$ both in backward and forward direction. Their intersections indicate a closed stream line touching the boundary of the domain of $\mathbf{s}$. Figure 5.8b gives an illustration.


(a) LIC images at 3 different time slices.

(b) Closeup of a tracked closed stream line. The searching arcs and the resulting seeding line are shown as gray lines.

(c) Tracked critical points, Hopf bifurcations (green spheres), fold bifurcations (gray spheres).

(d) Saddle connections (red/blue flow ribbons), tracked closed stream lines (green surfaces).

Figure 5.9: Test data set: Stream line oriented topology of a 2D time-dependent vector field.

Cyclic fold bifurcations are the only type of the above listed events which we cannot extract without prior tracking of at least one of the two involved closed stream lines. Hence, our system is not able to detect closed stream lines which appear *and* disappear at cyclic folds.

5.2 Applications

We applied the previously described methods to a number of test data sets. Not surprisingly, not all topological features appear in all data sets, and different topological features turned out to be important for different data sets.

Figure 5.9 shows the topological visualization of a random 2D time-dependent data set on a $5 \times 5 \times 5$ grid. Random vector fields are useful tools for a proof-of-concept of topological methods, since they contain a maximal amount of topological information. Figure 5.9a shows LIC images of the vector field at three different time slices which already indicates a high topological complexity. Figure 5.9c shows parts of the topological skeleton. We detected 18 critical lines of $\mathbf{s}$ (shown in red/blue/yellow, according to their outflow/inflow/saddle behavior), 32 fold bifurcations (gray spheres), and 4 Hopf bifurcations (green spheres). In this figure we also included two LIC planes to illustrate the relation between the critical lines of **s** and the critical points of **v**. In addition, figure 5.9d shows 8 detected saddle connections, among them 4 periodic blue sky bifurcations. We visualized them as red/blue double flow ribbons describing the orientation of the intersecting separation surfaces which create them. Starting from the Hopf bifurcations, we tracked the closed stream lines of **s**: each closed stream line starting in a Hopf bifurcation turned out to end in a periodic blue sky bifurcation. The resulting surfaces are shown in green. Figure 5.9b shows a detail of figure 5.9d to illustrate the tracking of closed stream lines: also shown are the seeding arcs for each step of the integration.

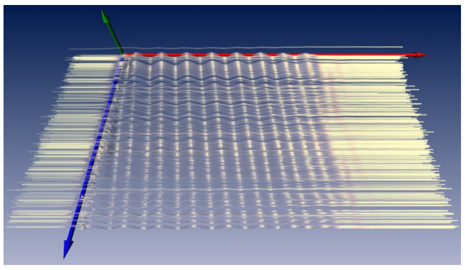
The computing time for extracting the saddle connections in this example was 20 seconds on a Pentium 4 1.7 GHz. For this, 42 attracting and 42 repelling stream surfaces had to be integrated and checked for intersections. For tracking the closed stream lines, our algorithm took 14 seconds on the same hardware. For this, 52 steps of the described algorithm had been carried out.

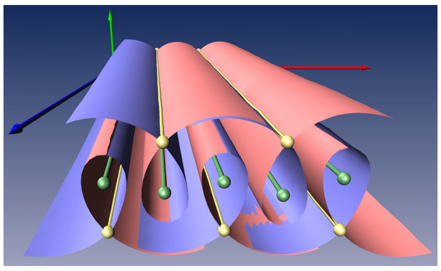
Figure 5.10 shows the visualization of a 2D time-dependent flow behind a circular cylinder. The cylinder is in the (x,y)-plane around the origin of the underlying coordinates system. This data set was kindly provided by Gerd Mutschke (FZ Rossendorf) and Bernd R. Noack (TU Berlin). Figure 5.10a shows the stream lines of **s** as illuminated lines [ZSH96]. As we can see in figure 5.10a, this incompressible flow does not contain critical points (except for a center and a saddle directly behind the cylinder). To make this data set applicable to a topological analysis, we subtract a constant vector field[4] which leads to the appearance of critical points. Figure 5.10c shows the topological skeleton of this modified data set which consists of 47 critical lines and 13 fold bifurcations. Note that the critical lines appear only in green and yellow, indicating that only moving centers and saddles are present. This corresponds to the fact that the vector field describes an incompressible flow. Figure 5.10c also shows that the critical points slowly move away from the cylinder over time: the critical lines of **s** are in general not parallel to the time axis. Figure 5.10b shows a close-up, where the separation surfaces emanating from the moving saddles are visualized. As we can see here, a number of separation surfaces tend to coincide making it impossible to extract isolated saddle connections. This is also due to the fact that the vector field is incompressible.

Figures 5.11 – 5.12 show the visualization of a vector field describing the flow over a 2D cavity. This data set was kindly provided by Mo Samimy and Edgar Caraballo (both Ohio State University) [CSJ] as well as Bernd R. Noack and Ivanka Pelivan (both TU Berlin). 1000 time steps have been simulated using the *compressible* Navier-Stokes equations; it exhibits a non-zero divergence inside the cavity, while outside the cavity the flow tends to have a quasi-divergence-free behavior. The topological skeleton has been extracted by dividing the temporal axis into 10 intervals and applying the extraction scheme onto each interval separately as shown in figure 5.11. The topological structures of the full data set visualized in figure 5.12a elucidate the quasi-periodic nature of the flow. Figure 5.12b shows approximately one period – 100 time steps – of the full data set, while figures 5.12c-d point out some topological details.

Figure 5.12b reveals the overall movement of the topological structures – the most dominating ones originating in or near the boundaries of the cavity itself. The quasi-divergence-free behavior outside the cavity is affirmed by the fact that a high number of Hopf bifurcations has been found in this area. The tracked closed stream line in

[4]This trick is well-known in the fluid dynamics community. It is motivated by the idea that the observer is moving with the flow.


(a) Stream lines of the autonomous system **s** (2.14) correspond to the stream lines of the 2D time-dependent flow field **v**. Time is denoted by the blue axis.


(b) Separation surfaces (close-up).

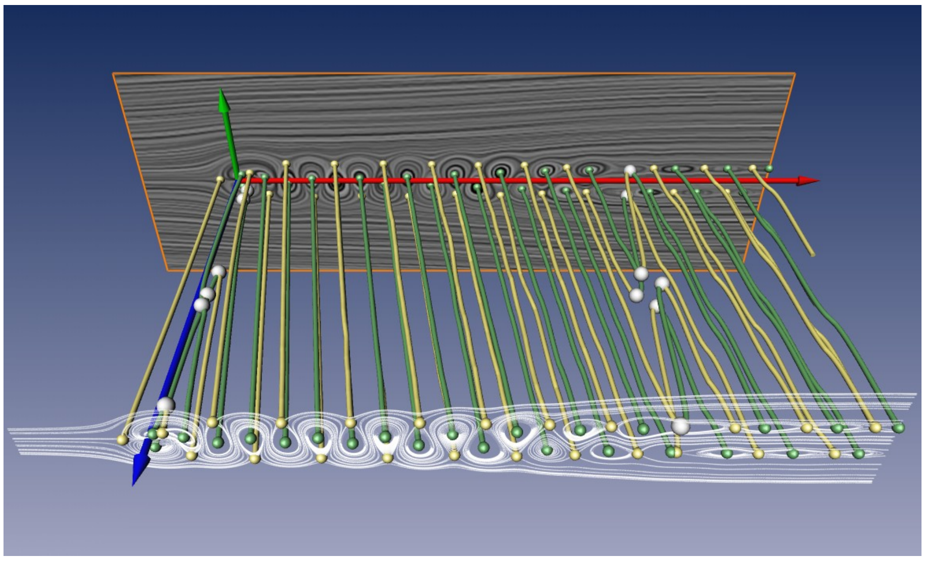

(c) Topological structures after subtracting a constant vector field.

Figure 5.10: 2D Flow behind a circular cylinder.

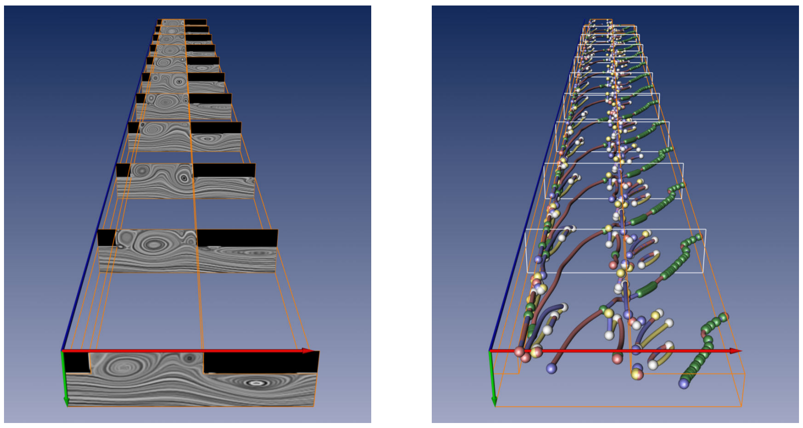
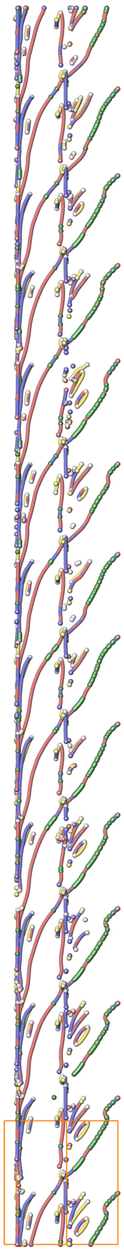
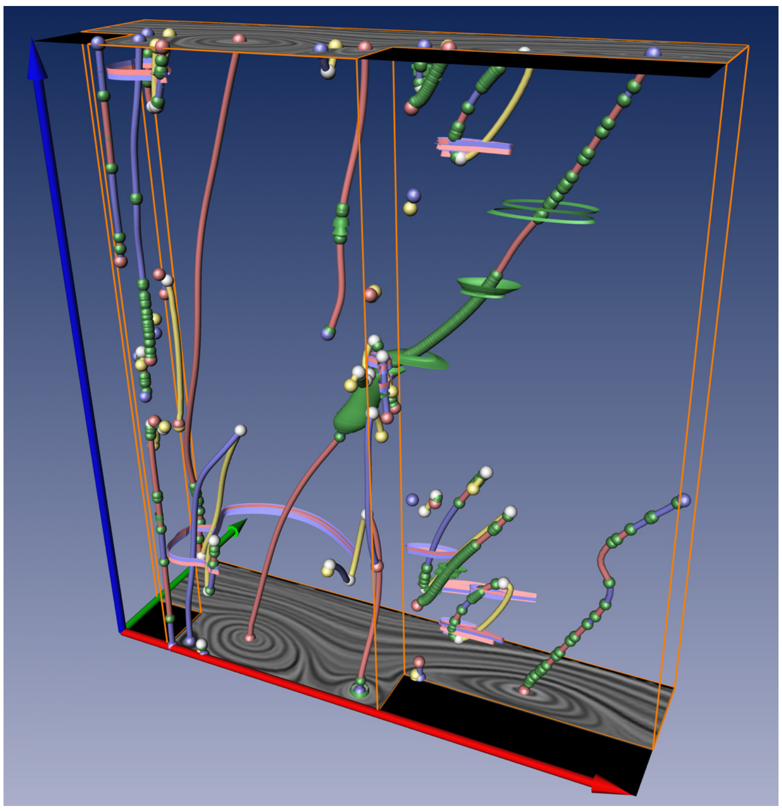
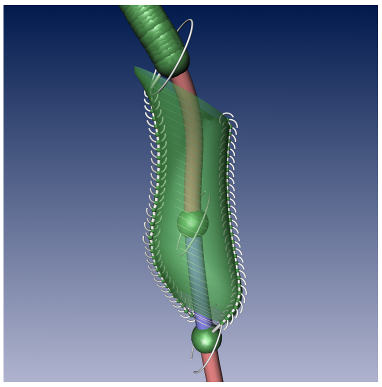


Figure 5.11: Cavity data set consisting of 1000 time steps. The temporal axis has been subdivided into the 10 depicted sections (left) in order to track the critical points (right).


(a) 1000 time steps.


(b) Topological skeleton of the first 100 time steps.


(c) Tracked closed stream line starting and ending in a Hopf bifurcation. Tracking from both sides yields the same result as shown by the searching arcs $\hat{\mathbf{k}}$ (section 5.1.2).

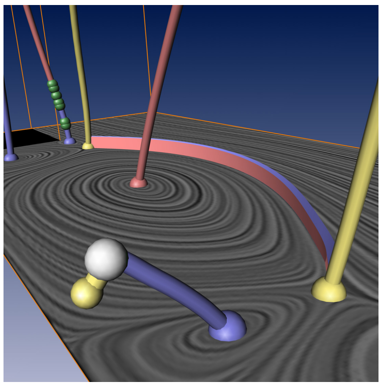
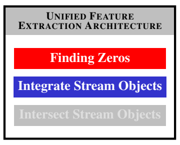

(d) Detail view with a saddle connection and a fold bifurcation.

Figure 5.12: 2D time-dependent flow at a cavity. The dataset consists of 1000 time steps which have been visualized in (a). All other images show the first 100 time steps.

figure 5.12c starts in a Hopf bifurcation and ends in another one – thereby enclosing a third Hopf. Our algorithm tracked it starting from both sides. The searching arcs $\hat{\mathbf{k}}$ as well as both resulting seeding lines $[\mathbf{x}_0, \ldots, \mathbf{x}_n]$ have been visualized to show the paths that have been taken by our method. Figure 5.12d shows a detailed view of time step 22, where a saddle connection has been detected. In the front of this figure a sink is going to join and disappear with a saddle, which just happened to enter at the domain boundary.

5.3 Two-Parameter-Dependent Topology



We aim at extracting all occurrences of critical points in two-parameter-dependent 2D vector fields $\mathbf{v}(x,y,s,t)$ where x,y represent the spatial and s,t denote the additional dimensions of the domain. In fact, s can be considered as the scale-space parameter while t serves as the physical time parameter, but other interpretations of s,t are possible as well. Similar to the one-parameter-dependent case we need to apply the algorithms for finding zeros and for integrating stream objects. The presented extraction scheme relies on the concept of Feature Flow Fields.

For the further explanation of the concepts, we distinguish two domains of $\mathbf{v}$: the 2D spatial subdomain D with fixed s and t, and the full 4D domain $\widetilde{D}$. For the sake of simplicity, we consider $D = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$ and mention that our algorithms work for more general domain boundaries as well. Furthermore, we choose $\widetilde{D} = D \times [s_{min}, s_{max}] \times [t_{min}, t_{max}]$. In the following we write $\mathbf{v}, \mathbf{w}, \ldots$ for 2D vector fields but $\widetilde{\mathbf{v}}, \widetilde{\mathbf{w}}, \ldots$ for 4D vector fields in $\widetilde{D}$.

As already discussed in section 2.3.2, the locations of the critical points of $\mathbf{v}$ build surface structures in $\widetilde{D}$. Different approaches to tracking critical points of $\mathbf{v}$ in $\widetilde{D}$ are possible. A Marching-Cubes- or Marching-Tetrahedra-like approach can be applied if the underlying grid provides a piecewise (quadri-)linear vector field. Bauer et al. use a similar approach to tracking 3D vortex core lines over time [BP02]. Here, we use a FFF-based approach ending up in a 4D stream surface integration. We do so, because the FFF approach is independent of an underlying grid, and it provides the tool to detect and classify bifurcations in $\widetilde{D}$ (section 5.3.2). Since the searched structures are surfaces in $\widetilde{D}$, we need two feature flow fields to track them. Following the ideas for tracking critical points in one-parameter-dependent flows (section 5.1.1) we use

$$\widetilde{\mathbf{f}} = \begin{pmatrix} \det(\mathbf{v}_y, \mathbf{v}_s) \\ \det(\mathbf{v}_s, \mathbf{v}_x) \\ \det(\mathbf{v}_x, \mathbf{v}_y) \\ 0 \end{pmatrix} \quad , \qquad \widetilde{\mathbf{g}} = \begin{pmatrix} \det(\mathbf{v}_y, \mathbf{v}_t) \\ \det(\mathbf{v}_t, \mathbf{v}_x) \\ 0 \\ \det(\mathbf{v}_x, \mathbf{v}_y) \end{pmatrix} . \tag{5.7}$$

$\widetilde{\mathbf{f}}$ tracks a critical point in s (keeping t constant), while $\widetilde{\mathbf{g}}$ tracks the critical points in t (keeping s constant). We give the following algorithm to get all seeding structures:

I Extract all critical points on the domain boundaries of $\widetilde{D}$, i.e., all points (x,y,s,t) with $[\mathbf{v}(x,y,s,t) = 0, E]$ and E is one of the 8 expressions $x = x_{min}$, $x = x_{max}$, $y = y_{min}$, $y = y_{max}$, $s = s_{min}$, $s = s_{max}$, $t = t_{min}$, $t = t_{max}$. Here, the approach for one-parameter-dependent vector fields can be applied.

II Extract all locations of fold bifurcations in $\widetilde{D}$. This can be done using the new FFF-based approach presented in the following section.

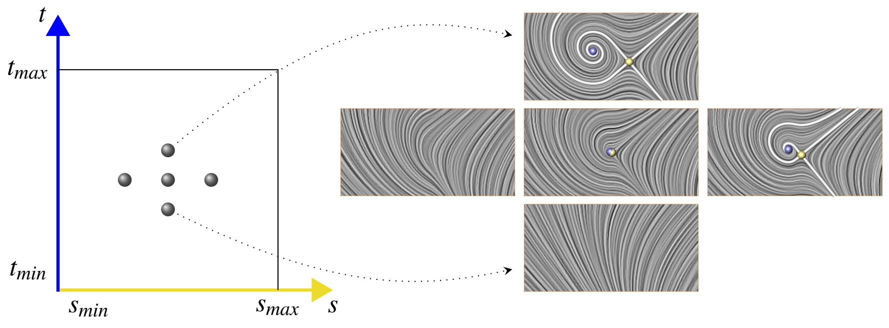


Figure 5.13: A (BB) fold bifurcation at $(\mathbf{x}_0, s_0, t_0)$: shown are the topological skeleton and the LIC images of $\mathbf{v}$ at (s_0, t_0) and $(s_0 \pm \varepsilon, t_0 \pm \varepsilon)$. The arrangement of the LIC images (right) corresponds to the points in the (s,t)-diagram (left). Starting from (s_0, t_0) moving forward either in s or in t creates a splitting of the critical point at (s_0, t_0).

This choice of seeding structures is a direct generalization of the one-parameter-dependent case. Both I and II yield line structures. Depending on whether one wants to track the critical points in s or in t one needs to apply a stream surface integration using $\widetilde{\mathbf{f}}$ or $\widetilde{\mathbf{g}}$ respectively. The union of both results yields all locations of all critical points of $\mathbf{v}$ in $\widetilde{D}$.

5.3.1 Tracking Fold Bifurcations

For one-parameter-dependent vector fields, fold bifurcations are isolated points serving as seeding structures to track the critical points. In two-parameter-dependent vector fields they build line structures in $\widetilde{D}$. Fold bifurcations occur at points in $\widetilde{D}$ fulfilling

$$[\,\mathbf{v} = \mathbf{0}\;,\;\det(\mathbf{v}_x, \mathbf{v}_y) = 0\,]\,. \tag{5.8}$$

Since $\widetilde{\mathbf{f}}.3 = \widetilde{\mathbf{g}}.4$ in (5.7), fold bifurcations occur simultaneously in s and in t. This means that depending on the signs of $(\nabla \widetilde{\mathbf{f}} \cdot \widetilde{\mathbf{f}}).3$ and $(\nabla \widetilde{\mathbf{g}} \cdot \widetilde{\mathbf{g}}).4$, a fold bifurcation in $\widetilde{D}$ can have 4 different BD classifications: (BB), (BD), (DB), (DD), where the first letter denotes the BD classification in s and the second does so in t. Figure 5.13 illustrates a (BB) fold bifurcation. To track the locations of fold bifurcations in $\widetilde{D}$, we apply the FFF approach again. Its two parts are explained in the following.

FFF for Tracking Fold Bifurcations

Let $\widetilde{\mathbf{h}}$ be the 4D vector field in question to track fold bifurcations in $\widetilde{D}$. Using the abbreviation

$$d = \det(\mathbf{v}_x, \mathbf{v}_y), \tag{5.9}$$

$\widetilde{\mathbf{h}}$ has to point into the direction where $\mathbf{v}$ and d remain constant in a local first order approximation. This means that $\widetilde{\mathbf{h}}$ has to fulfill

$$[\,\widetilde{\mathbf{h}} \perp \nabla u \quad , \quad \widetilde{\mathbf{h}} \perp \nabla v \quad , \quad \widetilde{\mathbf{h}} \perp \nabla d\,] \tag{5.10}$$

with

$$\nabla u = \begin{pmatrix} u_x \\ u_y \\ u_s \\ u_t \end{pmatrix} \quad , \quad \nabla v = \begin{pmatrix} v_x \\ v_y \\ v_s \\ v_t \end{pmatrix} \tag{5.11}$$

$$\nabla d = \begin{pmatrix} d_x \\ d_y \\ d_s \\ d_t \end{pmatrix} = \begin{pmatrix} \det(\mathbf{v}_{xx},\mathbf{v}_y) + \det(\mathbf{v}_x,\mathbf{v}_{xy}) \\ \det(\mathbf{v}_{xy},\mathbf{v}_y) + \det(\mathbf{v}_x,\mathbf{v}_{yy}) \\ \det(\mathbf{v}_{xs},\mathbf{v}_y) + \det(\mathbf{v}_x,\mathbf{v}_{ys}) \\ \det(\mathbf{v}_{xt},\mathbf{v}_y) + \det(\mathbf{v}_x,\mathbf{v}_{yt}) \end{pmatrix} . \tag{5.12}$$

Equation (5.10) describes a linear system which gives a unique solution for $\widetilde{\mathbf{h}}$ (except for scaling):

$$\widetilde{\mathbf{h}} = \begin{pmatrix} \det(\nabla u \ , \ \nabla v \ , \ \nabla d \ , \ \mathbf{i}_1) \\ \det(\nabla u \ , \ \nabla v \ , \ \nabla d \ , \ \mathbf{i}_2) \\ \det(\nabla u \ , \ \nabla v \ , \ \nabla d \ , \ \mathbf{i}_3) \\ \det(\nabla u \ , \ \nabla v \ , \ \nabla d \ , \ \mathbf{i}_4) \end{pmatrix} \tag{5.13}$$

where $\mathbf{i}_1,..,\mathbf{i}_4$, are the columns of the 4×4 unit matrix. Equation (5.13) can be rewritten as

$$\widetilde{\mathbf{h}} = \begin{pmatrix} -d_y \det(\mathbf{v}_s,\mathbf{v}_t) + d_s \det(\mathbf{v}_y,\mathbf{v}_t) - d_t \det(\mathbf{v}_y,\mathbf{v}_s) \\ +d_x \det(\mathbf{v}_s,\mathbf{v}_t) - d_s \det(\mathbf{v}_x,\mathbf{v}_t) + d_t \det(\mathbf{v}_x,\mathbf{v}_s) \\ -d_x \det(\mathbf{v}_y,\mathbf{v}_t) + d_y \det(\mathbf{v}_x,\mathbf{v}_t) - d_t\, d \\ +d_x \det(\mathbf{v}_y,\mathbf{v}_s) - d_y \det(\mathbf{v}_x,\mathbf{v}_s) + d_s\, d \end{pmatrix} . \tag{5.14}$$

Starting Points for Integrating $\widetilde{\mathbf{h}}$

In order to compute the starting points for integrating $\widetilde{\mathbf{h}}$, we have to compute the intersection points of the paths of the fold bifurcations with the 8 boundary surfaces of $\widetilde{D}$, i.e., locations with

$$[\,(5.8)\ ,\ E\,] \tag{5.15}$$

and E is one of the 8 expressions $x = x_{min}$, $x = x_{max}$, $y = y_{min}$, $y = y_{max}$, $s = s_{min}$, $s = s_{max}$, $t = t_{min}$, or $t = t_{max}$. Solving each of those 8 systems is equivalent to finding isolated zeros in a 3D vector field.

In addition, we have to extract inner bifurcation points where fold bifurcations appear or disappear. We call them *fold-fold bifurcations* and treat them in the next section.

5.3.2 Fold-fold Bifurcations

A fold-fold bifurcation is the event of collapsing and disappearing of two fold bifurcations while moving forward either in s or in t (or the reverse process: the appearance and splitting of two fold bifurcations). Fold-fold bifurcations occur at points with vanishing third or fourth component of $\widetilde{\mathbf{h}}$.

There are three kinds of structurally stable fold-fold bifurcations. A *s-fold-fold* bifurcation is characterized by

$$\left[(5.8) \ , \ \widetilde{\mathbf{h}}.3 = 0 \ , \ \widetilde{\mathbf{h}}.4 \neq 0 \right] . \tag{5.16}$$

A *t-fold-fold* bifurcation is characterized by

$$\Big[(5.8) \ , \ \ \widetilde{\mathbf{h}}.3 \neq 0 \ \ , \ \ \widetilde{\mathbf{h}}.4 = 0 \Big] . \tag{5.17}$$

A *s-t-fold-fold* bifurcation is characterized by

$$\Big[(5.8) \ , \ \ \widetilde{\mathbf{h}}.3 = 0 \ \ , \ \ \widetilde{\mathbf{h}}.4 = 0 \Big] . \tag{5.18}$$

In order to show that structurally stable solutions in $\widetilde{D}$ for all three kinds of bifurcations exist, we note that (5.8) implies that $\mathbf{v}_x$ and $\mathbf{v}_y$ are parallel, i.e., $\mathbf{v}_y = \lambda\, \mathbf{v}_x$ for a certain λ. Inserting this and $d = 0$ into (5.14) gives

$$\begin{aligned} \widetilde{\mathbf{h}}.3 &= -(\lambda\ d_x - d_y)\ \det(\mathbf{v}_x, \mathbf{v}_t) \\ \widetilde{\mathbf{h}}.4 &= \quad (\lambda\ d_x - d_y)\ \det(\mathbf{v}_x, \mathbf{v}_s). \end{aligned} \tag{5.19}$$

This means that an s-fold-fold bifurcation occurs at $[\,(5.8)\ ,\ \det(\mathbf{v}_x, \mathbf{v}_t) = 0\,]$ which is equivalent to

$$[\ \mathbf{v} = \mathbf{0}\ \ ,\ \ \widetilde{\mathbf{g}} = \mathbf{0}\ \ ] . \tag{5.20}$$

A t-fold-fold bifurcation occurs at $[\,(5.8)\ ,\ \det(\mathbf{v}_x, \mathbf{v}_s) = 0\,]$ which is equivalent to

$$\Big[\ \mathbf{v} = \mathbf{0}\ \ ,\ \ \widetilde{\mathbf{f}} = \mathbf{0}\ \Big] . \tag{5.21}$$

An s-t-fold-fold bifurcation occurs at $[\,(5.8)\ ,\ \lambda\ d_x - d_y = 0\,]$ which is equivalent to

$$[\ \mathbf{v} = \mathbf{0}\ \ ,\ \ d_y\, \mathbf{v}_x = d_x\, \mathbf{v}_y\ \ ] . \tag{5.22}$$

s-fold-fold and t-fold-fold bifurcations can be further classified by applying well-known approaches from vector field topology: an analysis of the eigenvalues of the Jacobian at the critical point. For an s-fold-fold, we consider the eigenvalues of $\nabla\widetilde{\mathbf{g}}$ at the bifurcation. They have the form $0, 0, -\sqrt{r_s}, \sqrt{r_s}$ where r_s is a certain real number. This can be shown by inserting $\mathbf{v}_y = \lambda \mathbf{v}_x$ and $\mathbf{v}_t = \mu \mathbf{v}_x$ into the first order partials of $\widetilde{\mathbf{g}}$. This yields

$$\widetilde{\mathbf{g}}_x = \begin{pmatrix} \det(\mathbf{v}_x, \lambda \mathbf{v}_{xt} - \mu \mathbf{v}_{xy}) \\ \det(\mathbf{v}_x, \mu \mathbf{v}_{xx} - \mathbf{v}_{xt}) \\ 0 \\ \det(\mathbf{v}_x, \mathbf{v}_{xy} - \lambda \mathbf{v}_{xx}) \end{pmatrix} \qquad \widetilde{\mathbf{g}}_y = \begin{pmatrix} \det(\mathbf{v}_x, \lambda \mathbf{v}_{yt} - \mu \mathbf{v}_{yy}) \\ \det(\mathbf{v}_x, \mu \mathbf{v}_{xy} - \mathbf{v}_{yt}) \\ 0 \\ \det(\mathbf{v}_x, \mathbf{v}_{yy} - \lambda \mathbf{v}_{xy}) \end{pmatrix}$$
$$\widetilde{\mathbf{g}}_s = \begin{pmatrix} \det(\mathbf{v}_x, \lambda \mathbf{v}_{st} - \mu \mathbf{v}_{ys}) \\ \det(\mathbf{v}_x, \mu \mathbf{v}_{xs} - \mathbf{v}_{st}) \\ 0 \\ \det(\mathbf{v}_x, \mathbf{v}_{ys} - \lambda \mathbf{v}_{xs}) \end{pmatrix} \qquad \widetilde{\mathbf{g}}_t = \begin{pmatrix} \det(\mathbf{v}_x, \lambda \mathbf{v}_{tt} - \mu \mathbf{v}_{yt}) \\ \det(\mathbf{v}_x, \mu \mathbf{v}_{xt} - \mathbf{v}_{tt}) \\ 0 \\ \det(\mathbf{v}_x, \mathbf{v}_{yt} - \lambda \mathbf{v}_{xt}) \end{pmatrix} . \tag{5.23}$$

The matrix $\nabla\widetilde{\mathbf{g}} = (\widetilde{\mathbf{g}}_x, \widetilde{\mathbf{g}}_y, \widetilde{\mathbf{g}}_s, \widetilde{\mathbf{g}}_t)$ has rank 2, because for the third line of $\nabla\widetilde{\mathbf{g}}$ we have $\nabla\widetilde{\mathbf{g}}.3 = (0,0,0,0)$, and for the remaining lines holds $\nabla\widetilde{\mathbf{g}}.1 + \lambda\, \nabla\widetilde{\mathbf{g}}.2 + \mu\, \nabla\widetilde{\mathbf{g}}.4 = (0,0,0,0)$. This gives two eigenvalues of 0. Furthermore, the remaining eigenvalues add to 0, since $\mathrm{trace}(\nabla\widetilde{\mathbf{g}}) = 0$. This yields the eigenvalue structure $0, 0, -\sqrt{r_s}, \sqrt{r_s}$. In a similar way it can be shown that $\nabla\widetilde{\mathbf{f}}$ has the eigenvalues $0, 0, -\sqrt{r_t}, \sqrt{r_t}$ at a t-fold-fold bifurcation.

Considering the eigenvalues of $\nabla\widetilde{\mathbf{g}}$ we get the following classification for a s-fold-fold:

	s-fold-fold	t-fold-fold	s-t-fold-fold
saddle	$\widetilde{\mathbf{f}} \neq \mathbf{0}, \widetilde{\mathbf{g}} = \mathbf{0}$ $r_s > 0$	$\widetilde{\mathbf{f}} = \mathbf{0}, \widetilde{\mathbf{g}} \neq \mathbf{0}$ $r_t > 0$	$d_y \mathbf{v}_x = d_x \mathbf{v}_y$
closed collapse	$\widetilde{\mathbf{f}} \neq \mathbf{0}, \widetilde{\mathbf{g}} = \mathbf{0}$ $r_s < 0$	$\widetilde{\mathbf{f}} = \mathbf{0}, \widetilde{\mathbf{g}} \neq \mathbf{0}$ $r_t < 0$	

Table 5.1: The five cases of fold-fold bifurcations.

- $r_s < 0$ gives a *closed collapse bifurcation:* while moving forward/backward in s, a closed zero-line of $\mathbf{v}$ becomes smaller, collapses to a point, and disappears.
- $r_s > 0$ gives a *saddle bifurcation:* while moving forward/backward in s, two branches of the zero-curves of $\mathbf{v}$ move toward each other until they intersect and make the two fold bifurcations disappear.

The case $r_s = 0$ gives a structurally unstable event and is not considered here.

The classification of a t-fold-fold bifurcation into saddle and closed collapse works in a similar way. Here we compute the eigenvalues of $\nabla\widetilde{\mathbf{f}}$ which turn out to have the structure $0, 0, -\sqrt{r_t}, \sqrt{r_t}$ for a certain real number r_t. Then $r_t > 0$ gives a saddle bifurcation and $r_t < 0$ gives a closed collapse bifurcation.

Figures 5.14a–5.14c illustrate a number of fold-fold bifurcations. Figure 5.14a shows an s-fold-fold saddle bifurcation. The first row shows the visualization at 5 consecutive steps in s. Throughout the entire figure 5.14, the bifurcations occur in the third picture of every row. The same field as in the first row is shown in the second row of figure 5.14a, but now with interactively changing t. In fact, the third image of the first row and the third image of the second row are the visualizations at the same location (s,t) in which the bifurcation occurs. As we can see in the second row, no collapsing/splitting of the fold bifurcation occurs at an s-fold-fold bifurcation if moving in t. Figure 5.14b shows the same visualization for a t-fold-fold closed collapse bifurcation. Figure 5.14c shows an example of an s-t-fold-fold bifurcation. The two rows of this figure show that the splitting/collapsing of the fold bifurcation occurs while moving forward in both s and t.

Table 5.1 summarizes the conditions for the 5 possible cases of fold-fold bifurcations. Each of the cases occurs at structurally stable locations in $\widetilde{D}$ with $\mathbf{v} = \mathbf{0}$ and the additional conditions as shown in the table. A rather similar classification of moving line structures as considered here was obtained in [TSW$^+$05] in a different context: the tracking of vortex core lines in 3D time-dependent vector fields. Note that - similar to s-fold-fold and t-fold-fold bifurcations, the possible bifurcations of moving vortex core lines are also called saddle and closed collapse.

Each of the 5 fold-fold bifurcations can be further classified concerning their BD classification. In fact, an s-fold-fold and a t-fold-fold can be either a birth or a death event of two fold bifurcations. This means that the BD classification of an s-fold-fold and a t-fold-fold bifurcation is either (B) or (D). For an s-t-fold-fold the BD classification can be considered for both s and t, yielding 4 cases for the BD classification: (BB), (BD), (DB), (DD). This means that by adding a BD classification to the 5 fold-fold bifurcations, we end up with 12 different cases of fold-fold bifurcations. For example, the complete classification of the fold-fold bifurcations in figure 5.14a is (D)-s-fold-fold saddle, figure 5.14b shows a (B)-t-fold-fold closed collapse, and figure 5.14c shows a

s-fold-fold	*t*-fold-fold	*s*-*t*-fold-fold
(BB) $\leftrightarrow$ (BD)	(BB) $\leftrightarrow$ (DB)	(BB) $\leftrightarrow$ (DD)
(DB) $\leftrightarrow$ (DD)	(BD) $\leftrightarrow$ (DD)	(BD) $\leftrightarrow$ (DB)

Table 5.2: Classification of fold-fold bifurcations wrt. the BD classification of the collapsing/splitting fold bifurcations.

(DB)-*s*-*t*-fold-fold.

Fold-fold bifurcations can also be classified from another point of view: from the BD classification of the two collapsing/splitting fold bifurcations. If for instance a (BB) fold bifurcation collapses with a (BD) fold, we write (BB) $\leftrightarrow$ (BD) for the fold-fold bifurcation. Table 5.2 shows the relation to the above-mentioned classification. Note that this classification is different to the BD classification of fold-fold bifurcations mentioned above. Here, we consider the BD classifications of the collapsing fold bifurcations, while in the other case we consider the BD classification of the fold-fold bifurcations themselves.

5.3.3 Tracking Hopf Bifurcations

A Hopf bifurcation, i.e., an event where a spiraling source changes to a spiraling sink or vice versa, occurs at

$$[\,\mathbf{v} = \mathbf{0}\,,\ \mathrm{div}(\mathbf{v}) = 0\,] \tag{5.24}$$

together with $\det(\mathbf{v}_x, \mathbf{v}_y) > 0$. Similar to fold bifurcations, Hopf bifurcations build line structures in $\widetilde{D}$. For tracking them, we use again the FFF approach. In fact, we track all parts fulfilling (5.24) and subsequently eliminate all parts of the resulting lines with $\det(\mathbf{v}_x, \mathbf{v}_y) \leq 0$. As part of the tracking, we consider the events of collapsing/splitting two Hopf bifurcations: the Hopf-fold bifurcations.

FFF for Tracking Hopf Bifurcations

Let $\widetilde{\mathbf{k}}$ be the FFF in question for tracking Hopf bifurcations. Using the abbreviation

$$e = \mathrm{div}(\mathbf{v}) = u_x + v_y, \tag{5.25}$$

$\widetilde{\mathbf{k}}$ has to fulfill

$$[\,\widetilde{\mathbf{k}} \perp \nabla u \quad , \quad \widetilde{\mathbf{k}} \perp \nabla v \quad , \quad \widetilde{\mathbf{k}} \perp \nabla e\,]. \tag{5.26}$$

which is a linear system with a unique solution for $\widetilde{\mathbf{k}}$ (except for scaling):

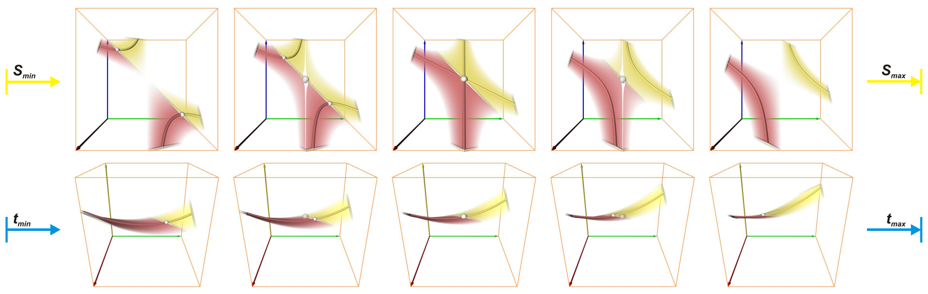
$$\widetilde{\mathbf{k}} = \begin{pmatrix} \det(\ \nabla u\ ,\ \nabla v\ ,\ \nabla e\ ,\ \mathbf{i}_1\ ) \\ \det(\ \nabla u\ ,\ \nabla v\ ,\ \nabla e\ ,\ \mathbf{i}_2\ ) \\ \det(\ \nabla u\ ,\ \nabla v\ ,\ \nabla e\ ,\ \mathbf{i}_3\ ) \\ \det(\ \nabla u\ ,\ \nabla v\ ,\ \nabla e\ ,\ \mathbf{i}_4\ ) \end{pmatrix}. \tag{5.27}$$

Note that the last two components of $\widetilde{\mathbf{k}}$ are

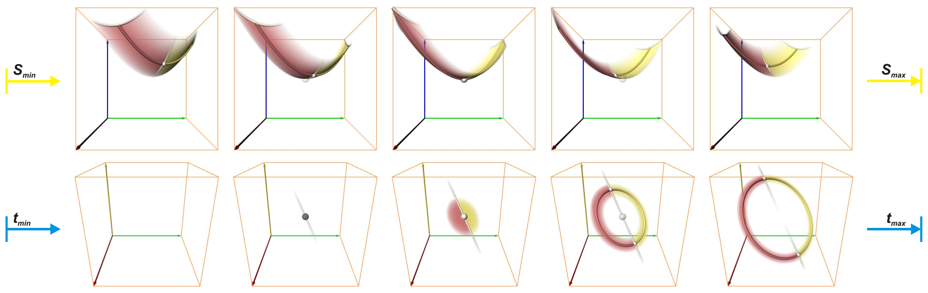
$$\widetilde{\mathbf{k}}.3 \quad = \quad -e_x \det(\mathbf{v}_y, \mathbf{v}_t) + e_y \det(\mathbf{v}_x, \mathbf{v}_t) - e_t\, d \tag{5.28}$$

$$\widetilde{\mathbf{k}}.4 \quad = \quad +e_x \det(\mathbf{v}_y, \mathbf{v}_s) - e_y \det(\mathbf{v}_x, \mathbf{v}_s) + e_s\, d. \tag{5.29}$$

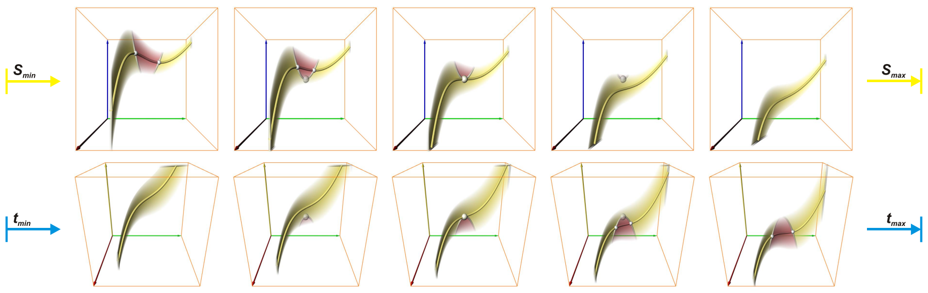
The starting points for integrating $\widetilde{\mathbf{k}}$ are the Hopf bifurcations at the boundary of $\widetilde{D}$ as well as the Hopf-fold bifurcations, which will be treated in the following.



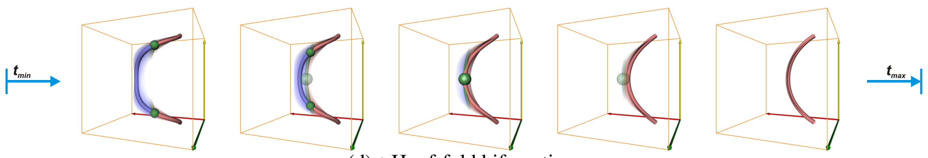
(a) *s*-fold-fold saddle bifurcation.



(b) *t*-fold-fold closed collapse bifurcation.



(c) *s*-*t*-fold-fold bifurcation.



(d) *t*-Hopf-fold bifurcation.

Figure 5.14: Fold-fold and Hopf-fold bifurcations.

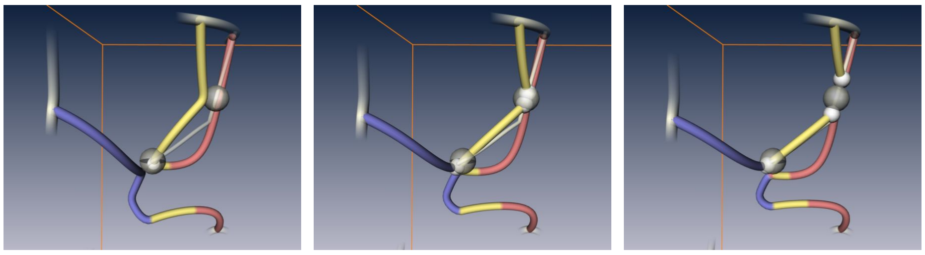

(a) before (b) the bifurcations (c) after

Figure 5.15: Noise data set: two saddle bifurcations.

Hopf-fold Bifurcations

There are two kinds of Hopf-fold bifurcations. A *s-Hopf-fold* is the event of collapsing two Hopf bifurcations while moving forward/backward in s. They occur at structurally stable isolated points with

$$[\,(5.24)\,,\,\widetilde{\mathbf{k}}.3 = 0\,]. \tag{5.30}$$

Their detection is equivalent to finding zeros in 4D vector fields. At a *t-Hopf-fold* bifurcation, two Hopf bifurcations collapse while moving forward/backard in t. The condition for them is

$$[\,(5.24)\,,\,\widetilde{\mathbf{k}}.4 = 0\,]. \tag{5.31}$$

Figure 5.14d illustrates a Hopf-fold bifurcation.

5.4 Applications

In this section we extract the topological skeletons of a number of two-parameter-dependent vector fields.

Figures 5.15–5.16 show the visualization of a random data set on a 4^4 grid. Random vector fields are used as a proof-of-concept since they contain the maximal amount of topological information. Figure 5.16 shows an overview in (x,y,s)-visualization. We see the paths of the critical points at a certain time t and at the boundaries of $\widetilde{D}$. Figure 5.15 shows a detail of the data: two s-fold-fold saddle bifurcations in (x,y,s)-visualization.

Figures 5.18–5.17 show the visualization of a 2D time-dependent vector field describing the flow behind a 2D cylinder. The data set was kindly provided by Gerd Mutschke (FZ Rossendorf) and Bernd R. Noack (TU Berlin). This flow exhibits periodic vortex shedding leading to the well known von Kármán vortex street. This phenomenon plays an important role in many industrial applications, like mixing in heat exchangers or mass flow measurements with vortex counters. However, this vortex shedding can lead to undesirable periodic forces on obstacles, like chimneys, buildings, bridges and submarine towers. The original version of this non-compressible flow does not contain critical points (except for two directly behind the cylinder). The right-hand image of figure 5.17a illustrates it at a certain time t. In order to make topological methods applicable, a constant vector field can be subtracted which corresponds to an observer moving with the flow. In order to study the influence of the

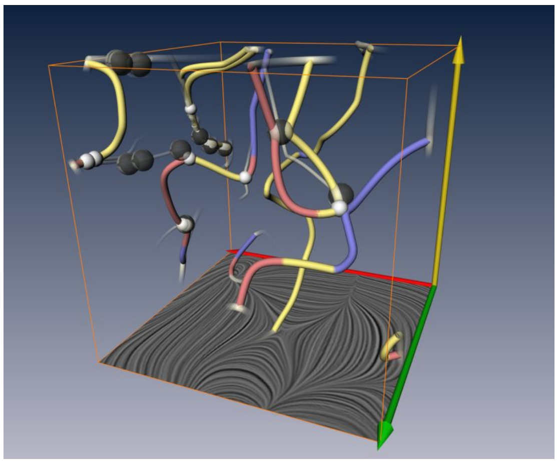
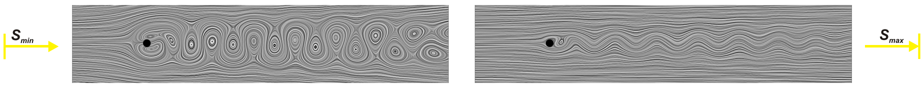
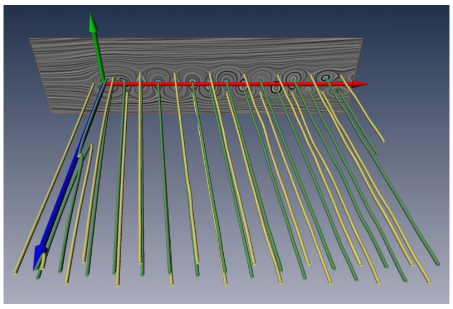


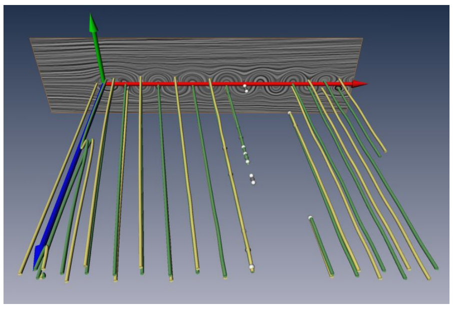
Figure 5.16: Noise data set: overview.



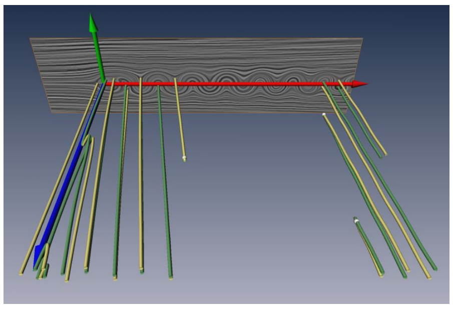
(a) LIC image for a certain t at $s = 0.15$ and $s = 1$



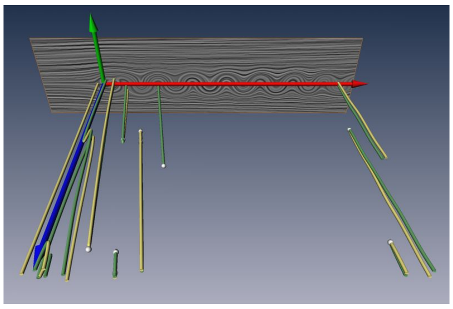
(b) s=0.15



(c) s=0.6



(d) s=0.8



(e) s=1

Figure 5.17: Flow behind a circular cylinder: s controls the subtracted constant field, t is the physical time parameter. (a) LIC image for a certain t and two different s, (b)–(e) (x, y, t)-visualization for 4 different s.

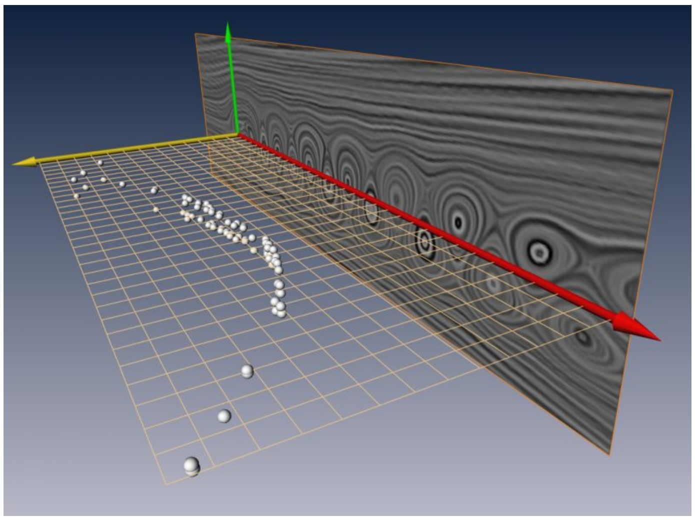
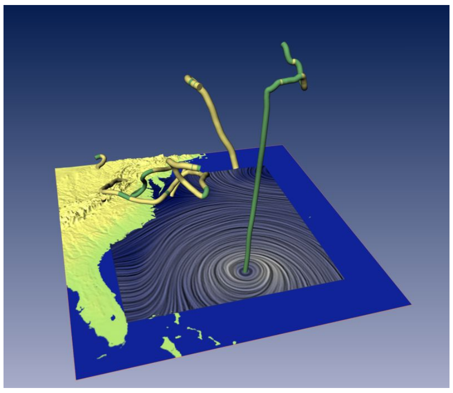
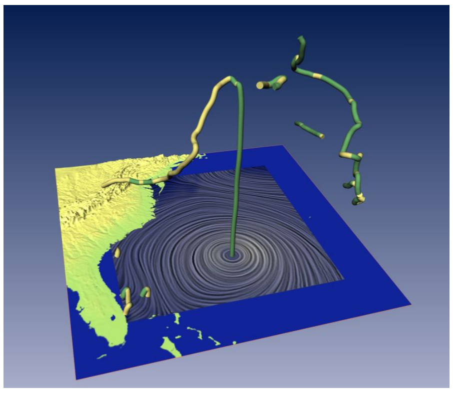


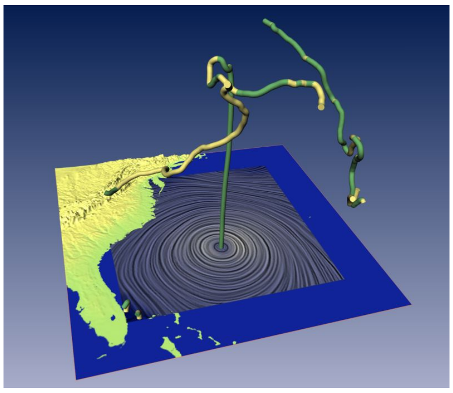
Figure 5.18: Cylinder flow: all fold-fold bifurcations in (x,y,s)-visualization.

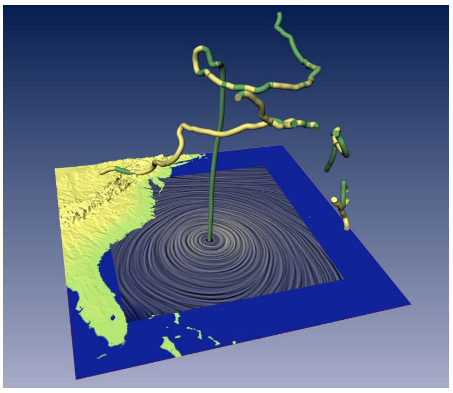
subtracted part, we apply our topological analysis: if the average flow of the original field is approximately $(1,0)^T$ as for the given data set, we subtract the constant vector field $(1-s)\cdot(1,0)^T$ with $s \in [0,1]$. This gives the two-parameter-dependent vector field $\mathbf{v}(x,y,s,t) = \mathbf{v}(x,y,t) - (1-s)\cdot(1,0)^T$. Figure 5.17a already shows for one particular t that increasing s from 0 to 1 will make the critical points collapse and disappear. Figures 5.17b–5.17e show $\mathbf{v}(x,y,t)$ at 4 different s-values. Each of the images shows the moving critical points in t as yellow (saddle) and green (center) 3D curves. The larger s gets, the fewer moving critical points are present. Figures 5.17b–5.17e also show that the disappearance of the moving critical points does not start from one of the boundaries of D but at a certain area located downstream of the cylinder. To analyze this further, we switch to a (x,y,s)-visualization as shown in figure 5.18. Here we visualized all detected fold-fold bifurcations. We can clearly see the parabola-like shape whose minimum denotes the location where – when moving forward in s – the first moving critical points disappear. For this data set we detected 92 fold-fold bifurcations, all of them are s-fold-fold saddle bifurcations. The computing time was in the range of several hours for detecting the paths of the critical points of $\mathbf{v}$ at the boundaries of $\widetilde{D}$. However, if this computation is carried out once, the extraction of a skeleton for a particular s/t is possible in less than a second on a current PC based system.

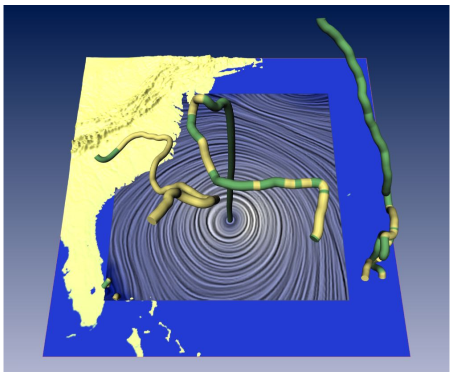
Figure 5.19 shows the visualization of the wind components of the Hurricane Isabel data set from the IEEE Visualization 2004 contest. It was produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR and the U.S. National Science Foundation (NSF). Although this is a one-parameter-dependent 3D vector field $\mathbf{v}(x,y,z,t) = (u,v,w)^T$, the main flow takes place in u- and v-direction. By neglecting the w-component and setting $z = s$, we obtain a two-parameter-dependent 2D field from which we analyze the first half of the original 48 time steps. Figures 5.19a–5.19d show the (x,y,s)-visualization for different t-steps. We can clearly see the dominating moving critical point (green line) changing its location while moving forward in t. Also, for high s-values the dominating critical point tends to have a fold bifurcation with a moving saddle (yellow). Figure 5.19e shows the top view at t=19. Figure 5.19f shows the location of the dominating critical point of $\mathbf{v}$ at the boundary $s = s_{min}$ of $\widetilde{D}$ (white line). Note that this line serves as the seeding structure to integrating the moving critical points at different t-values.


(a) t=4


(b) t=14


(c) t=20


(d) t=25


(e) t=19 (top view)

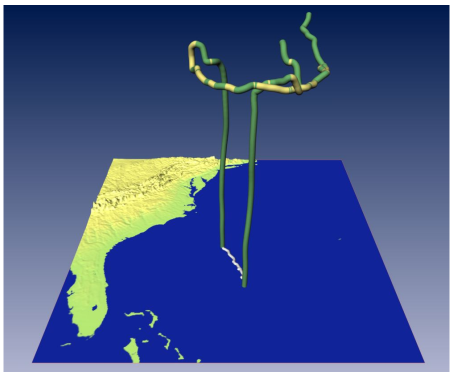

(f) dominating critical point at t=4, t=25, and at the boundary of $\widetilde{D}$ (white line)

Figure 5.19: (x,y,s)-visualization of the Hurricane Isabel data set.

Chapter 6

Extraction of Higher Order Topological Structures and Topological Simplification

In this chapter we present an approach to extracting higher order critical points of 3D vector fields which can be used as a topological simplification method. Although extraction of higher order critical points [SKMR98] and topological simplification [TSH00, TSH01a] are well-researched for 2D vector fields, we are not aware of any topological simplification or higher order critical point extraction approaches for 3D vector fields. In fact, Laramee et al. found in their state of the art report [LHZP07] that the method presented here is the first one in the visualization and computer graphics community to address topological simplification for 3D vector fields.

Our solution can be considered as a 3D extension of [TSH00]. There, a convex closed polygon is placed around an area of interest (e.g. a cluster of critical points) and the potential location **c** of the higher order critical point is set inside the polygon. Then a segmentation into areas of different flow behavior is achieved by analyzing number and order of the points **x** on the polygon where the vector of the vector field is either parallel or perpendicular to $\mathbf{x} - \mathbf{c}$.

Similar to [TSH00], our solution for a 3D vector field **v** first assumes that the location of a higher order critical point is known. Then a closed convex surface **s** with genus zero is placed around **c**, and the topological classification of **c** is achieved by analyzing the flow behavior towards **c** for stream lines starting from each point of **s**. The result is a segmentation of **s** into areas of different flow behavior around **c**. For this, an icon is created which represents the higher order critical point in the visualization.

Although this approach primarily assumes that **s** surrounds a very small area around a critical point, it can also be used to cover larger areas, for instance clusters of critical points. In this case, the location of the critical point **c** has to be set inside **s** – for example at the average position of all critical points inside **s**. Then the result is an icon for a higher order critical point which replaces the complete topological skeleton of the area inside **s** in the visualization. This way the algorithm acts as a topological simplification technique.

It can also be seen as an extension of [MR02] which puts a closed surface around a region of interest as well in order to compute the index of the circumscribed area by a geometric algebra approach. However, our approach does not focus on the index of a critical point but on segmentation of areas of different flow behavior. Also, [MR02] mainly aims at the detection of critical points while our approach allows to identify their type and is applied to simplify 3D vector fields.

An approach to a continuous topology simplification of 2D vector fields is presented in [TSH01a]. Based on the graph structure of the topological skeleton and certain relevance measures, pairs of critical points are removed by local changes to the vector values at the grid nodes. We are not aware of any extension to 3D. A major obstacle to such an undertaking seems to be the computation of the graph structure of a 3D vector field, since separation surfaces come into play here.

6.1 Extracting the Sectors Around a Critical Point

In this section we describe how to segment **s** into areas of different F-classification. We first assume that we know the location of a higher order critical point **c** in **v**, such that a small closed surface **s** can be placed around it. However, in practice higher order critical points rarely appear because they are usually split into clusters of first order critical points. Our approach works for these cases as well: **s** is placed around the cluster, and in addition the location of **c** has to be set inside **s**. To do so, we either choose the average position of the critical in the cluster, or we simply choose the center of **s**. In fact, our approach can be applied to *any* region of **v** giving a valid topological description of it.

The main contribution of this section is to show that the segmentation of **s** into areas of different F-classification essentially corresponds to the extraction of the topological skeleton of an appropriate 2D vector field **u** on **s** where every critical point of **u** is provided with an additional Bit of information. The segmentation of **s** is done entirely by sampling **v** on **s**: the behavior of **v** outside **s** is not considered. This approach is reflected by the introduction of a new 3D vector field **w** which is a simplified version of **v**: **v** and **w** coincide on **s** but may differ in other areas. We start with the introduction of **u** and **w** as well as some other auxiliary vector fields.

6.1.1 Auxiliary vector fields

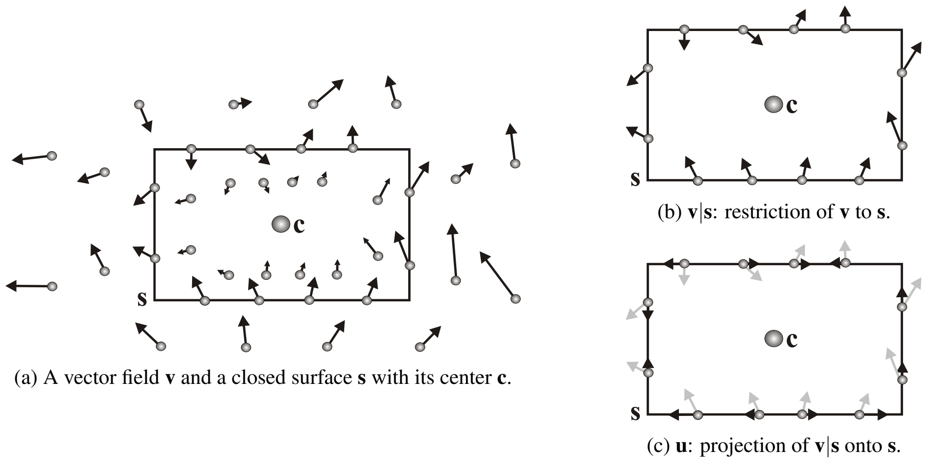
Again, we consider a 3D vector field **v**, a closed surface **s** in the domain of **v**, and a point **c** inside **s**. Here we use an axis-parallel box which simplifies the following implementations: vector operation on **s** become operations on a collection 2D plane vector fields. Figure 6.1a gives an illustration of a 2D example. Out of this, we define a number of auxiliary vector fields:

- Let $\mathbf{v}|\mathbf{s}$ be the restriction of **v** to the domain **s**. This means that $\mathbf{v}|\mathbf{s}$ is a map from the 2D domain **s** to $\mathbb{R}^3$. Figure 6.1b illustrates this. $\mathbf{v}|\mathbf{s}$ is the only part of **v** which is used to classify **c**.

- Let **n** be the normalized outward normal at each point of **s**.

- Let **u** be the central projection of $\mathbf{v}|\mathbf{s}$ onto **s** with the projection center **c**. For every point $\mathbf{x} \in \mathbf{s}$, $\mathbf{u}(\mathbf{x})$ can be computed as

$$\mathbf{u} = \alpha\,\mathbf{r} + \beta\,\mathbf{v} \tag{6.1}$$

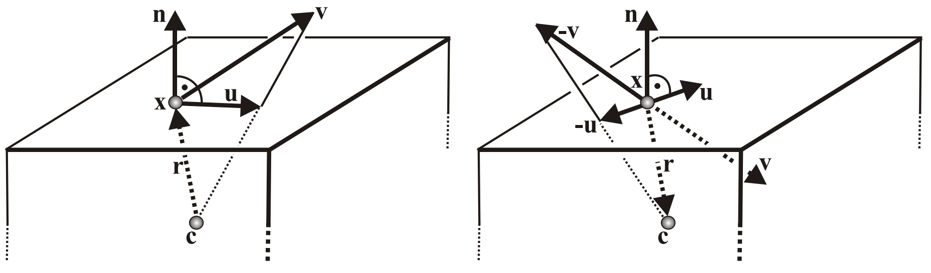


(a) A vector field **v** and a closed surface **s** with its center **c**.

(b) **v**|**s**: restriction of **v** to **s**.

(c) **u**: projection of **v**|**s** onto **s**.

Figure 6.1: The original vector field **v** is restricted to and central projected onto a closed surface **s**. This yields the auxiliary vector fields **v**|**s** and **u**.



(a) for $\mathbf{n}\,\mathbf{v} \geq 0$.

(b) for $\mathbf{n}\,\mathbf{v} < 0$.

Figure 6.2: Obtaining the central projection vector field **u**.

with

$$\mathbf{r} = \operatorname{sign}(\mathbf{n}\,\mathbf{v}) \cdot (\mathbf{x} - \mathbf{c}), \quad \alpha = \frac{-(\mathbf{n}\,\mathbf{v})}{(\mathbf{n}\,\mathbf{r}) + (\mathbf{n}\,\mathbf{v})}, \quad \beta = \frac{(\mathbf{n}\,\mathbf{r})}{(\mathbf{n}\,\mathbf{r}) + (\mathbf{n}\,\mathbf{v})}. \qquad (6.2)$$

Figure 6.2 explains this. Figure 6.1c illustrates **u** for the 2D example.

- Let **w** be constructed in the following way: given a point $\mathbf{x} \in \mathbb{R}^3$ with $\mathbf{x} \neq \mathbf{c}$, we intersect **s** with the line segment $(\mathbf{x}, \mathbf{c})$. This gives a unique intersection point $\mathbf{s}_\mathbf{x}$ on **s** which can be written as $\mathbf{s}_\mathbf{x} = (1 - \lambda_\mathbf{x})\,\mathbf{c} + \lambda_\mathbf{x}\,\mathbf{x}$ for a certain $\lambda_\mathbf{x} > 0$. Then we compute $\mathbf{w}(\mathbf{x}) = \mathbf{v}(\mathbf{s}_\mathbf{x})/\lambda_\mathbf{x}$. In addition we set $\mathbf{w}(\mathbf{c}) = (0,0,0)^T$. Figure 6.3a illustrates the definition of **w**.

Note that **u** is a 2D vector field on the surface **s**, while **v** and **w** are 3D vector fields. **w** can be considered as a simplified version of **v**: **v** and **w** are identical on **s**, while all other vectors of **w** are obtained by a linear interpolation between **c** and a point on **s**. Since we only use **v**|**s** for the classification, we are allowed to do the classification for **w** instead for **v**.

Due to the construction of **u** and **w**, there are a number of relations between them: $\mathbf{x} \in \mathbf{s}$ is a critical point of **u** iff $\mathbf{v}(\mathbf{x}) = \gamma(\mathbf{x} - \mathbf{c})$ for a certain $\gamma \neq 0$. In this case, the

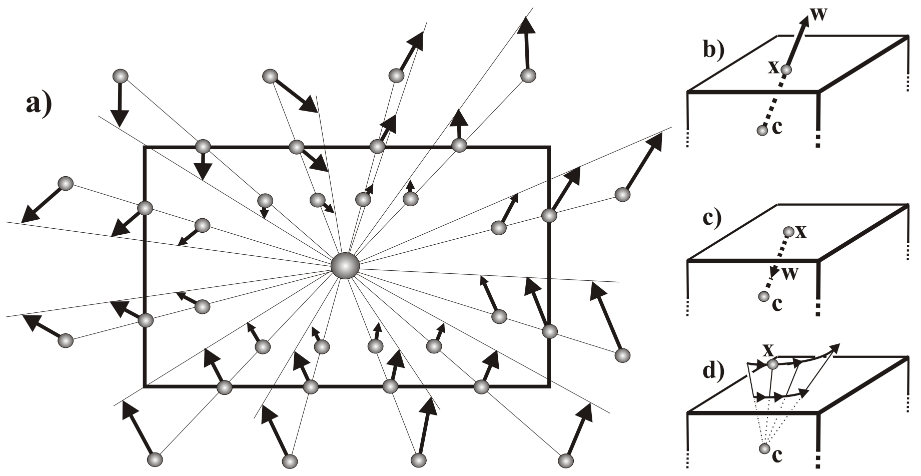


Figure 6.3: a) Auxiliary 3D vector field **w**; b) **x** is a **w**-outflow critical point of **u**; c) **x** is **w**-inflow critical point of **u**; d) the central projection of a stream line of **w** is a stream line of **u**.

stream line of **w** starting in **x** is a straight line from **x** to **c**, either in forward or in backward integration. Hence, every critical point of **u** can be classified by two criteria:

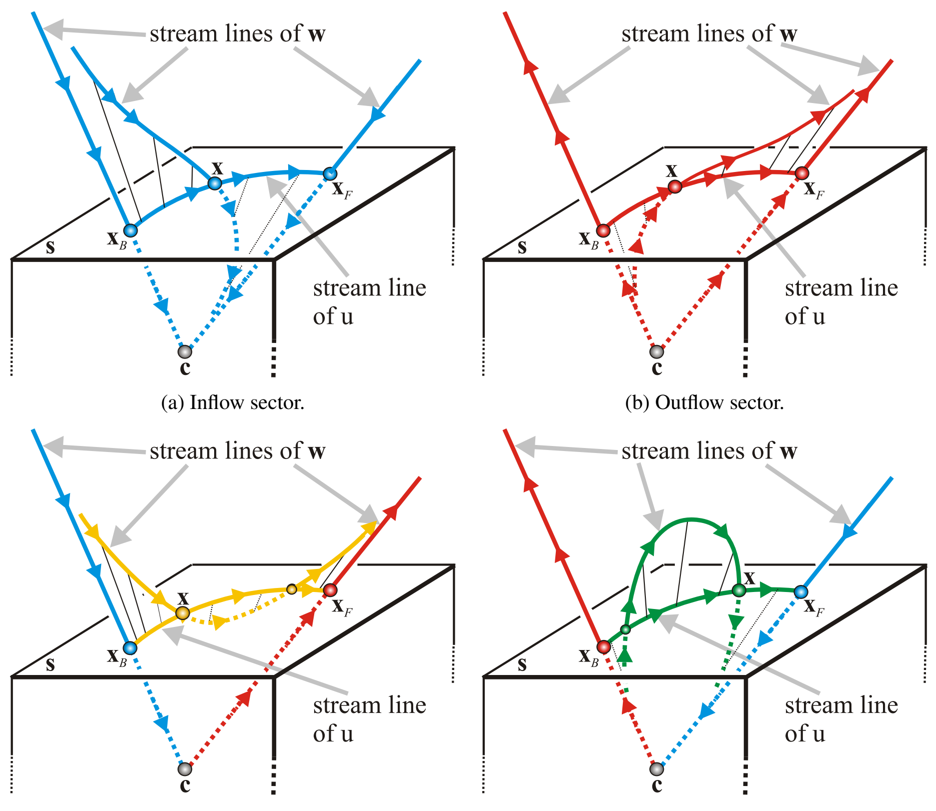
- concerning the classical topological classification of a critical point of a 2D vector field, i.e., as a source, sink, or saddle. (The structurally unstable case of a center is not considered here.)
- concerning the sign of γ: **x** is called a **w**-inflow point if $\gamma < 0$, i.e., $\mathbf{w}(\mathbf{x})$ points to **c**. For $\gamma > 0$, **x** is called a **w**-outflow point. Figures 6.3b and 6.3c illustrate this.

Another relation between **u** and **w** can be established concerning their stream lines. Let $\mathbf{x} \neq \mathbf{c}$ be a point in the 3D domain of **w**. Considering the stream line of **w** starting in **x**, it turns out that its central projection onto **s** is a stream line of **u**. This follows directly from the definition of **u** and **w**. Figure 6.3d illustrates this relation.

6.1.2 F-classification of a single point on s

After establishing the relations between **u** and **w**, we can get the F-classification of an arbitrary point $\mathbf{x} \in \mathbf{s}$ in a simple way. Instead of integrating **w** starting from **x** and observing whether or not the stream line ends in **c**, we apply a 2D stream line integration of **u** starting from **x**. This integration ends in two certain critical points of **u**: $\mathbf{x}_B$ by backward and $\mathbf{x}_F$ by forward integration. Usually $\mathbf{x}_B$ is a source and $\mathbf{x}_F$ is a sink, but they might be saddles as well – in case that **x** lies on a separatrix of the topological skeleton of **u**. However, the F-classification of **x** can simply be obtained by considering the **w**-inflow/outflow behavior of $\mathbf{x}_B$ and $\mathbf{x}_F$ respectively, as summarized in table 6.1 and illustrated in figure 6.4.

Our goal is to segment **s** into areas of different F-classification. Since we are now able to F-classify any point on this surface, we could naively do so for a high number of sample points. In figure 6.5 we did this for a test data set (a region in the benzene data set, which will be explained in detail in section 6.2 – we use this test data set throughout the rest of this section for algorithmic explanations). It clearly shows distinctive areas



(a) Inflow sector. (b) Outflow sector.

(c) Hyperbolic sector. (d) Elliptic sector.

Figure 6.4: F-classification of a single point $\mathbf{x}$ on $\mathbf{s}$ depends on the $\mathbf{w}$-inflow/outflow behavior of the end points of the stream line through $\mathbf{x}$ in $\mathbf{u}$.

end point $\mathbf{x}_B$ backward integration	end point $\mathbf{x}_F$ forward integration	F-classification of $\mathbf{x}$	
$\mathbf{w}$-inflow	$\mathbf{w}$-inflow	inflow	🟦
$\mathbf{w}$-outflow	$\mathbf{w}$-outflow	outflow	🟥
$\mathbf{w}$-inflow	$\mathbf{w}$-outflow	hyperbolic	🟨
$\mathbf{w}$-outflow	$\mathbf{w}$-inflow	elliptic	🟩

Table 6.1: F-classification of a single point $\mathbf{x}$ on $\mathbf{s}$. Same color coding as given in table 2.2 on page 39.

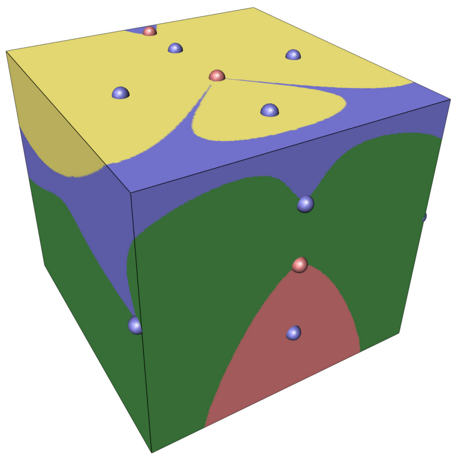


Figure 6.5: Naive approach of F-classifying **s**: 256^2 points on each face have been F-classified according to table 6.1 by integrating stream lines starting/ending in the depicted sources/sinks. Computation time: 700 seconds (P4 3.4GHz).

of color coded flow behavior. There are three major reasons, why this approach is not satisfying:

- It takes a rather long time to integrate the stream lines for a reasonable resolution. In this example, we started a stream line integration at 256^2 sample points on each face of the box. This took more than 700 seconds on a Pentium 4 with 3.4 GHz.

- It only yields a visual representation of the different sectors. One still needs to apply some other algorithm in order to get a feature-based separation, i.e., lines or points.

- It does not capture all features on **s**. Even already for first order 3D saddles enclosed by **s** there are points and lines on the surface with a different F-classification than their surrounding area (hyperbolic area vs. inflow/outflow points and lines). Even for very high resolutions, such points and lines are only hit by accident.

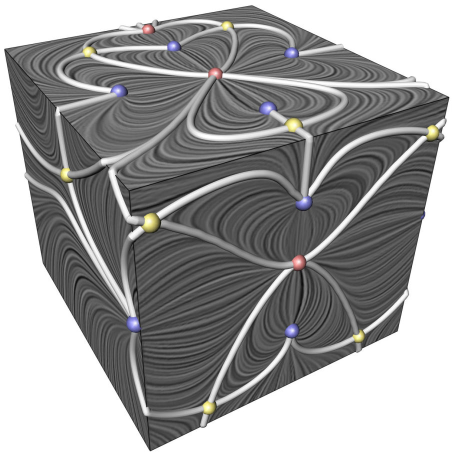
In the following subsections we present an algorithm which captures all features of **w** and is orders of magnitude faster.

6.1.3 F-classification of all points on s

For the F-classification of a single point, we utilized the stream line of **u** through that point and looked at the **w**-inflow/outflow behavior of its start and end point. Those start and end points are critical points of **u**. It is easy to see, that all points **x** have the same F-classification, if all their stream lines start and end in the same pair of critical points. Thus, the topological skeleton of **u** gives the desired segmentation of **s**. We give the following algorithm:

1. Extract the topological skeleton of **u**. It consists of critical points and separation curves (Figure 6.6a).

2. F-classify all critical points of **u** concerning their **w**-inflow/outflow property (Figure 6.6b).


(a) Extracted skeleton of **u**.

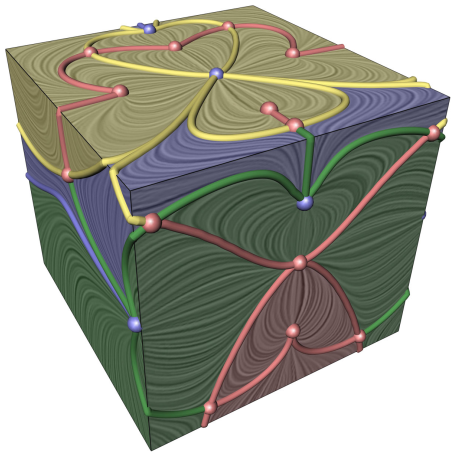

(b) F-classified skeleton of **u**.

Figure 6.6: Exploiting the topology of **u** to F-classify **s**.

3. F-classify all separatrices of **u**. To do so, consider the **w**-inflow/outflow property of the two critical points where the separatrix starts and ends (Figure 6.6b).

4. F-classify the remaining areas. To do so, consider the **w**-inflow/outflow property of the two critical points where the integration in forward and backward direction starts and ends (Figure 6.6b).

We describe details of this algorithm at an example. Figure 6.7a shows a certain substructure of a topological skeleton of **u** consisting of a source $\mathbf{x}_R$, a sink $\mathbf{x}_A$ and two saddles $\mathbf{x}_{Sa1}$, $\mathbf{x}_{Sa2}$. From each saddle, two separatrices have been integrated (one in forward and one in backward direction) which end in $\mathbf{x}_R$ and $\mathbf{x}_A$ respectively. In our example we assume that $\mathbf{x}_R$, $\mathbf{x}_A$ are **w**-outflow while $\mathbf{x}_{Sa1}$, $\mathbf{x}_{Sa2}$ are **w**-inflow, as shown in figure 6.7b. Note that the color coding of figure 6.7b is different than figure 6.7a: here the red color means **w**-outflow, and the blue color means **w**-inflow. Then the separatrices from the source $\mathbf{x}_R$ to both saddles have elliptic behavior, while the separatrices from both saddles to the sink $\mathbf{x}_A$ possess hyperbolic behavior (see table 6.1). All points *inside* the area surrounded by the four separatrices have the property that a forward integration of **u** starting from them ends in $\mathbf{x}_A$ while a backward integration ends in $\mathbf{x}_R$. Hence they are part of an outflow sector (table 6.1).

6.1.4 A complete system of 2D topological substructures

The example of figures 6.7a-b shows that the elements (i.e., critical points, separatrices, inner area) of a topological substructure may have a different F-classification. Since the F-classification is entirely based on the **w**-inflow/outflow behavior of the critical points, there are 16 cases of F-classifying this topological substructure with 4 critical points. Figure 6.8a illustrates this: the location of critical points is the same as in figure 6.7a for each of the cases. The color coding here corresponds to table 6.1 and gives the F-classification of the critical points, separatrices, and inner areas respectively.

Another possible topological substructure of **u** is shown in figure 6.7c: both outflow separatrices of a saddle $\mathbf{x}_{Sa}$ end in the same sink $\mathbf{x}_A$, whereas inside the surrounded area there is a source with a separatrix to the saddle as well. A similar substructure

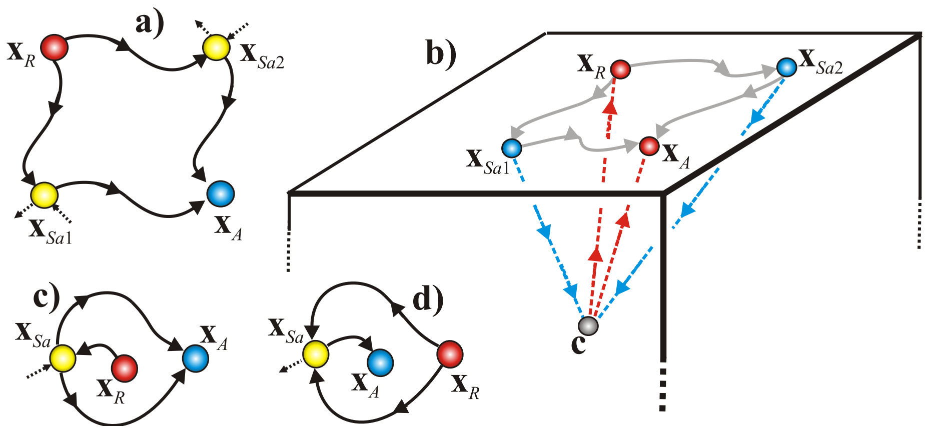
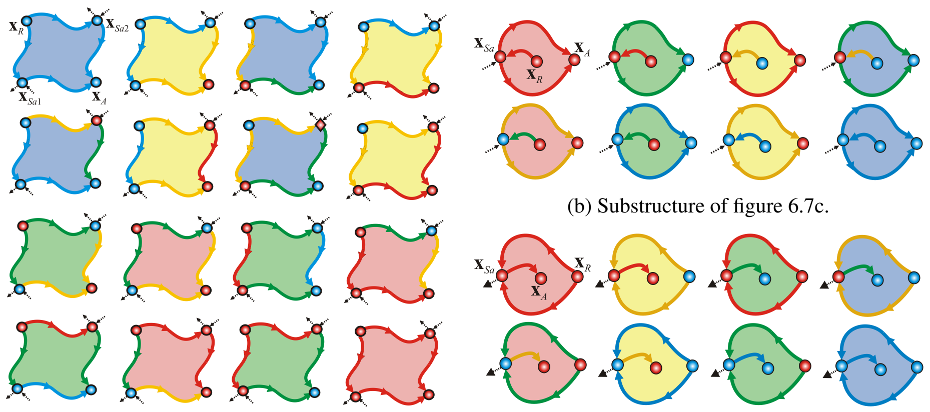


Figure 6.7: a) a topological substructure of $\mathbf{u}$ consisting of a source $\mathbf{x}_R$ (red), a sink $\mathbf{x}_A$ (blue), two saddles $\mathbf{x}_{Sa1}, \mathbf{x}_{Sa1}$ (yellow), and 4 separatrices; b) classification of the topological segment on $\mathbf{u}$ concerning the $\mathbf{w}$-inflow/outflow behavior of the critical points; c),d) other topological substructures of $\mathbf{u}$.



(a) Substructure of figure 6.7a.

(b) Substructure of figure 6.7c.

(c) Substructure of figure 6.7d.

Figure 6.8: Cases for F-classifying the substructures of the topological skeleton of $\mathbf{u}$. Colored according to table 6.1.

with exchanged source and sink is shown in figure 6.7d. Figures 6.8b-c show the F-classification of the 8 possible cases for each of those two types of substructures.

Finally we show that the topological substructures of figure 6.7 are essentially the only ones which can appear in **u**. We do so by considering the following facts:

- Each topological substructure consists of exactly one source and one sink. In fact, the inner area of a substructure describes exactly the area where the flow goes from this source to this sink. If there were for instance two sources, we would have another separation between them.

- Each separatrix goes from a saddle to a source or a sink. This means that we do not allow structurally unstable saddle connections (i.e., stream lines from one saddle to another).

- Each source/sink must not have more than two separatrices of the substructure ending in it. If a source/sink had three separatrices ending in it, the middle one would define another separation.

From these points it follows that the maximal number of separatrices involved in a substructure is 4. This gives that at most two saddles can be involved. The only case with two saddles is shown in figure 6.7a, the two cases with one saddle are shown in figures 6.7c and 6.7d. There is also a trivial case with no saddles involved: if **u** has only one source, one sink and no saddles, and the complete flow on **u** goes from the source to the sink. In this case, the complete area of **s** (except the critical points themselves) gets the F-classification concerning to the F-classification of the critical points.

6.1.5 Obtaining a minimal skeleton

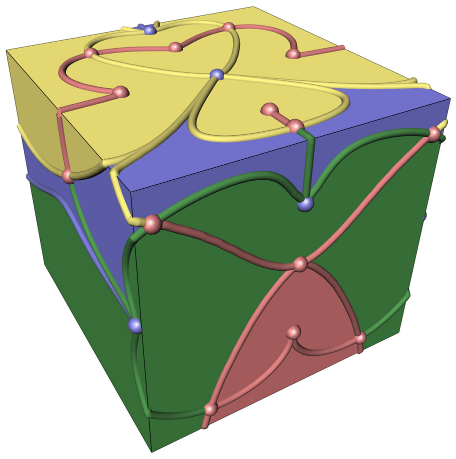
In section 6.1.3 we extracted the topological skeleton of the projected 2D vector field **u**. While these critical points and separation curves segment **u** into areas of different flow behavior, not all of them are necessarily needed to do the segmentation into areas of different F-classification. A redundancy with respect to the segmentation of **w** is introduced to that skeleton since only the **w**-inflow/outflow behavior has to be considered in order to do the segmentation. In other words, some neighboring substructures of the 2D skeleton may have the same F-classification and thus, the separation between them does not reflect different sectors of F-classification. Figures 6.9a-b illustrate this.

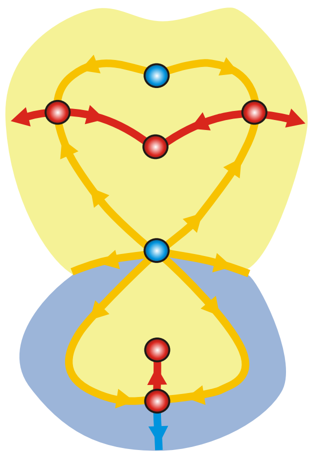
A minimal skeleton representing the different sectors of the F-classification only has to be found: structural elements of the 2D skeleton with identical behavior as their neighbors have to be either merged with them or completely removed. For this, we convert the 2D skeleton into a graph representation, where critical points reflect nodes, separation curves correspond to edges between two nodes, and inner areas are represented by their associated curves and points. We consider a graph to emphasize that any geometrical information can be discarded for the following. To minimize the skeleton, we have to

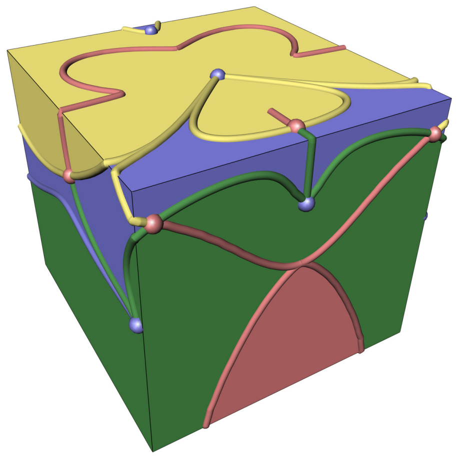
- remove edges, if they exhibit the same F-classification as their neighboring areas,

- merge areas, if an edge belonging to all of them has been removed,

- remove unconnected nodes (i.e., not connected to an undeleted edge), if they have the same F-classification as their surrounding area,


(a) Redundant skeleton.


(b) Substructures of the upper face.


(c) Minimal skeleton.

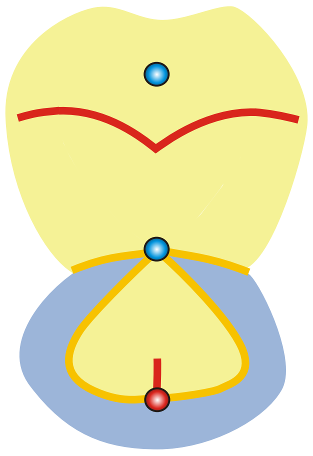
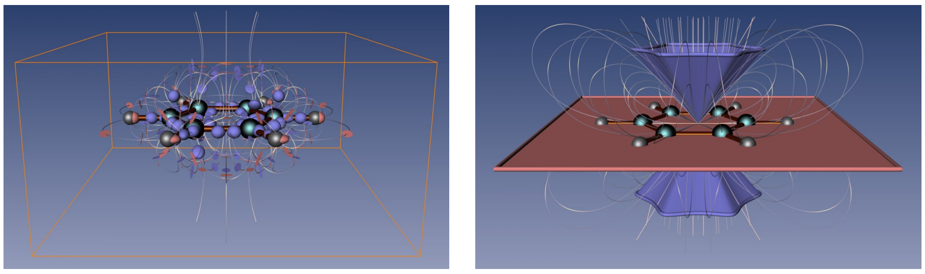

(d) Upper face after removal and merging.

Figure 6.9: F-classified topological skeleton of **u** before and after the removal of redundant elements.



(a) 184 first order critical points. The box around the molecule represents the chosen area for topological simplification.

(b) Topologically simplified representation with one higher order critical point elucidates the far field behavior of the benzene.

Figure 6.10: Topological representations of the electrostatic field of the benzene molecule.

- remove nodes, if they possess the same F-classification as all their connected edges,
- merge two edges, if both have been connected to the same deleted node.

Figures 6.9c-d illustrate the outcome of this process. Note that nodes cannot be merged since they correspond to isolated critical points in the skeleton, and areas can not be removed since their union covers the whole 2D domain on **s**. The obtained graph represents the minimal skeleton needed to distinguish between sectors of different F-classification. Critical points in this skeleton correspond to directions of straight inflow/outflow, while the remaining curves give the separation surfaces in between the different sectors (i.e., areas).

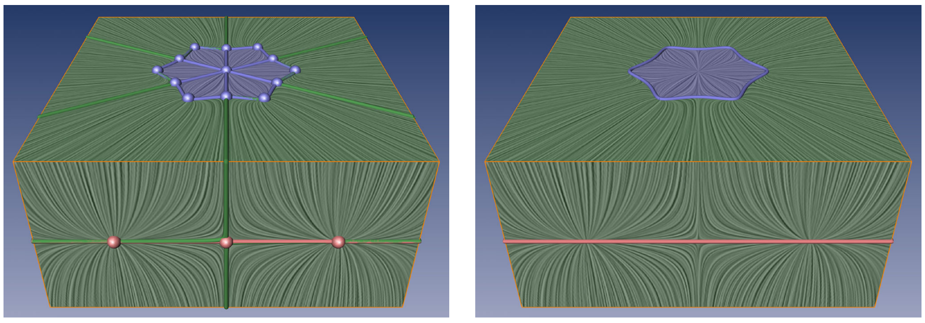
Our algorithm needed 4 seconds on our hardware to extract, F-classify and minimize the skeleton of the test data set. This is much faster than the naive approach depicted in figure 6.5. Furthermore, our algorithm guarantees to capture all features on **s**. For example, figure 6.9c shows in contrast to figure 6.5 an outflow separation curve between two hyperbolic areas on the upper face.

The resulting icon for the example used above is shown in figure 2.27b on page 42.

6.2 Applications

Figures 6.10 – 6.13 visualize the electrostatic field around a benzene molecule. This data set was calculated on a 101^3 regular grid using the fractional charges method described in [SS96]. Its topological richness is shown in figure 6.10a: it consists of 184 first order critical points.

This field describes the force of the electrostatic potential of the benzene molecule upon a positive point charge given in a certain location. If such a point charge is situated very close to the molecule, the closest atom will exert the highest force on it, i.e., attract or repel it. The influence of a single atom decreases the farther the point charge is located from the whole molecule. Instead, all atoms have nearly the same influence. One might say that the molecule as a whole is exerting force on a somewhat far located point charge. Thus, it is possible to distinguish between a near and a far field. Furthermore, sources and sinks of the electrostatic field represent minima and maxima of the potential. See [Bad90] for a further discussion of classifying atoms and molecules based on field topology.



(a) Extracted and F-classified skeleton. (b) Minimal skeleton.

Figure 6.11: Benzene data set: high level of topological abstraction.

These properties give a good setting for our algorithm. By placing a large box around the whole molecule, we yield a high level of abstraction. Figure 6.11a shows the box around the whole molecule together with the extracted and F-classified topological skeleton of the projected 2D vector field **u**. The minimal skeleton is depicted in figure 6.11b. As it can be seen here, there is a star-shaped inflow area (blue), an outflow line (red), and an elliptic area (green) between them in the visible parts of the box. Figure 6.10b shows the icon for this area together with stream lines of **w**. It clearly shows the behavior of the far field of the benzene molecule, if one compares it with figure 6.12.

In figure 6.13 we lowered the abstraction level by subdividing the domain in 3 (figure 6.13a) or 9 (figure 6.13b-c) subareas. This clearly shows the presence of a more complex topological behavior if we zoom into regions of interest. This is due to the fact that these detailed regions are governed by the near field because the influence of the individual atoms increases. Figure 6.13 shows that topologically rather complex structures are present which consist of complex areas of different F-classification.

The application of our technique to this topologically complex data set shows its usefulness at various levels of simplification: if a large area of interest is chosen, a rough global topological impression about the global behavior of the vector field can be obtained. Focusing the area of interest to particular smaller areas inside, topologically more complex structures become visible and provide a deeper insight into the topological behavior of the vector field.

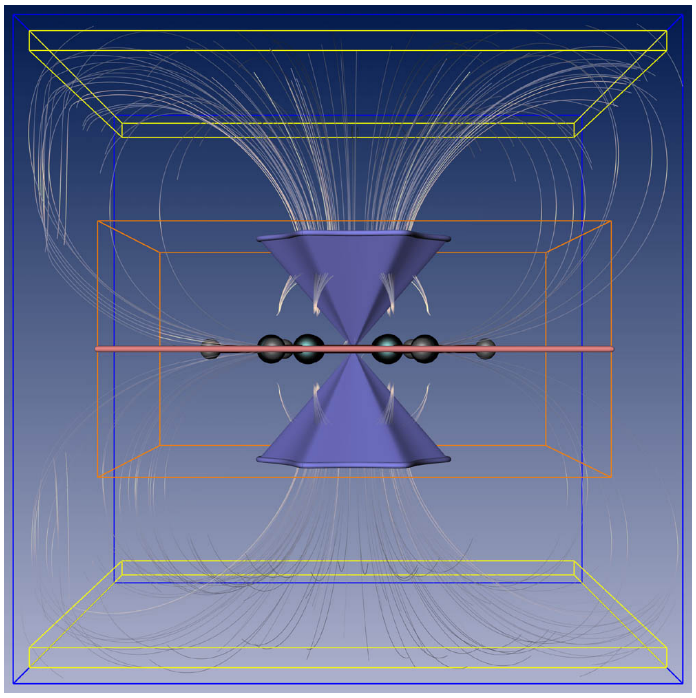
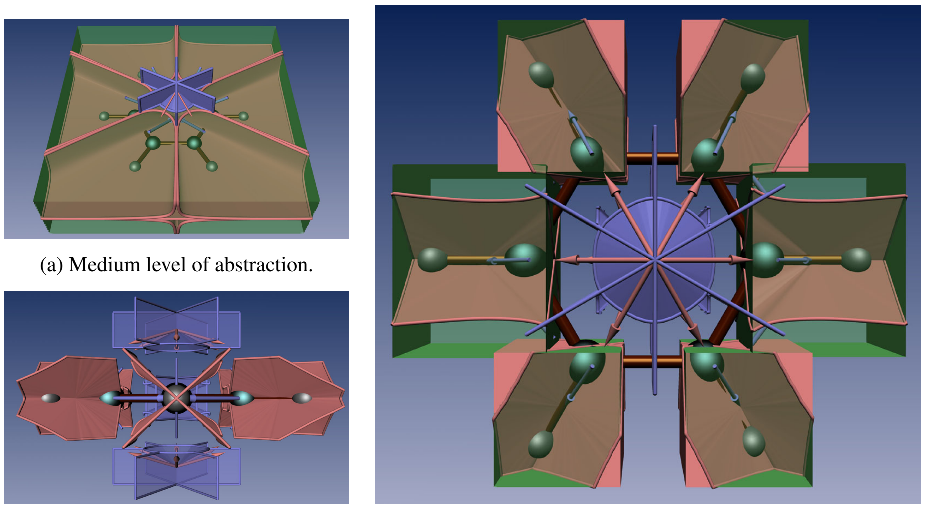


Figure 6.12: Benzene data set: High level of topological abstraction. Shown is the resulting higher order icon. Additionally, stream lines of the original vector field have been seeded inside the yellow boxes at the bottom and top.



(a) Medium level of abstraction.

(b) Low level of abstraction. View from side.

(c) Low level of abstraction. Front view.

Figure 6.13: Benzene data set: medium and low level of abstraction.

Chapter 7

Further Applications of Topological Concepts and Methods

In this chapter we show that topology has strong applications besides the structural depiction of the examined vector field.

The theoretical tools from chapter 2 will be utilized in section 7.1 to construct 3D vector fields from a given topological description consisting of points and lines. The result is a piecewise linear 3D vector field having the same topological entities as given by the design. This first approach to modeling 3D vector fields of arbitrary topology has been developed in the course of this thesis, and can serve as a foundation of a number of further applications.

The extraction methods developed in chapters 3 - 5 can also be used to extract vortex core lines as shown in section 7.2. Three different definitions of vortex core lines will be discussed and we show that all three types can be extracted using the Unified Feature Extraction Architecture. Furthermore, we give a unified notation of cores of swirling motion, present a novel way of describing vortex core lines as extremum lines, and link the fields of topology and vortex analysis by showing how vortex core lines can be described as topological separatrices of a derived vector field.

In section 7.3 we show how the results of a topological analysis can be used to steer other visualization techniques. Among other things, extracted features will be used to automate the seeding of stream lines and particles.

7.1 Construction of Higher Order 3D Vector Fields

While most of the vector fields to be visualized are obtained by simulation or measurement processes, Theisel presents an approach to *modeling* 2D vector fields of higher order topology, i.e., consisting of higher order critical points [The02]. This approach is based on two steps. First, the topological skeleton is interactively modeled by defining critical points and separation curves. Then a piecewise linear vector field is constructed which has the topological skeleton modeled before. The approach is also applied for a topology based compression technique.

In this section we show how to extend this approach to 3D. The methods in [The02] are strictly limited to 2D vector fields because of the following reasons:

- [The02] uses a complete segmentation of the areas around a 2D critical point into sectors of different flow behavior. Such a segmentation of 3D critical points did not exist prior to this thesis in the Visualization and Computer Graphics community. In fact, only first order critical points and the index of higher order critical points [MR02] had been considered before.
- Contrary to the 2D case, separatrices of 3D vector fields are particular stream surfaces. They tend to have a complex behavior even for rather simple vector fields (see section 2.2.4), making it a cumbersome (or even impossible) task to model them for instance as a parametric surface.
- There exists no approach to create a vector field which has a stream surface coinciding with a modeled parametric surface.

In this section we present the (to the best of our knowledge) first approach to modeling 3D vector fields of arbitrary topology. We extend the main ideas of [The02] to 3D and give solutions for the three problems mentioned above. This is based on the complete classification of 3D critical points into an arbitrary number of sectors of different flow behavior as given in section 2.4. To overcome the second and third problem, we adapt the concept of connectors (see section 3.2) by modeling not the separation surfaces themselves but only their intersection curves. The resulting algorithm is a two-step approach. First the user models a 3D topological skeleton, then a vector field is automatically constructed from this. For the user, the problem of modeling a vector field is reduced to the problem of modeling a topological skeleton by a number of control polygons.

We start with giving some motivation to create 3D vector fields by a modeling approach in the next section. In section 7.1.2 we introduce an approach to model the topological skeleton of a 3D vector field based on the already discussed theory of higher order critical points (section 2.4). Section 7.1.3 shows how to construct a vector field from a modeled skeleton. Finally, we demonstrate examples in section 7.1.4.

7.1.1 Why Modeling Vector Fields?

The common workflow in Scientific Visualization starts with an unknown data set, which mostly resulted from a simulation or an experiment. This data gets analyzed, visualized and interpreted to yield a higher understanding of the processes inherent to it. As an example, for the topological treatment of vector fields one would extract the topology from the data set and display it.

So why should one do the opposite by modeling a topological skeleton and constructing a vector field out of this? We believe that this approach can be applied in a number of situations.

Pattern Matching: One way of detecting features in vector fields is to create a reference vector field consisting of the desired feature, and compute its local similarity to the real data set [ES03, HEWK03]. The modeling approach may help to construct more involved patterns and therefore allow to search for more involved features in the data set.

Topology preserving compression: In most cases, the memory requirements for storing a topological skeleton are much smaller than for the vector field itself ([The02]).

This can be used for compression by just saving the topology and reconstructing a new vector field out of this later on. The construction methods presented here guarantee that all topological features of the original data set are contained in the compressed one as well.

Optimizing flow: The flow around an airfoil is subject to large efforts in order to increase the desired lift and to reduce the parasitic drag. These performance enhancements are achieved by changing the geometry (aerodynamic design) of the airfoil and controlling separation using air injection. Successful flow control strategies have mostly been based on a good physical understanding, which is more often achieved by a qualitative analysis as opposed to a quantitative one. Topology can aid this physical understanding not only by its visualization, but also by the ability to manipulate the original topology and to reconstruct the flow field out of this. This leads to a susceptibility study elucidating beneficial and detrimental changes to lift, drag and other parameters, because one can compute these measures for the reconstructed flow and compare it with the original ones. Thus, the beneficial topological structures for lift and drag can be identified. As a part of reverse engineering, actuation and sensing solutions for these beneficial structures can be developed. This computationally efficient topology-aided sensitivity analysis can guide heavy-duty numerical simulations and experimental setups.

Education and Testing: For the educational explanation of topological and other visualization methods, the algorithms described in this section can be used to create simple and illustrative data sets. Modeled vector fields may also serve as test data while developing new visualization techniques, but for their evaluation real data sets have to be used.

It is beyond the scope of this thesis to present solutions for all the applications mentioned above. Some of them may even lead to new research directions once a modeling approach is available.

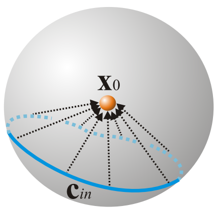
7.1.2 Modeling the Topological Skeleton

The idea of vector field topology is to separate regions of different flow behavior in the field. The main components of a topological skeleton are critical points and separation surfaces as discussed in chapter 2. In the following we describe how to interactively model them. Other topological features like boundary switch curves or closed stream lines are not considered here.

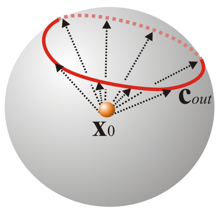
Critical Points

To model a critical point $\mathbf{x}_0$, we use the sphere model introduced in section 2.4, but in order to ease the modeling process we put a number of restrictions onto the separation curves on the sphere $\mathbf{s}$ around $\mathbf{x}_0$: all points on them have to have a F-classification of either inflow or outflow, and the separation curves have to be closed curves.

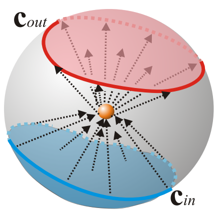
We detail this restriction in the following. Consider one inflow and one outflow surface of $\mathbf{x}_0$ defined by $\mathbf{c}_{in}$ and $\mathbf{c}_{out}$ on $\mathbf{s}$. Keeping in mind that $\mathbf{s}$ covers a very small neighborhood of $\mathbf{x}_0$, $\mathbf{c}_{in}$ and $\mathbf{c}_{out}$ must not intersect. They divide $\mathbf{s}$ into three different regions: the region inside $\mathbf{c}_{in}$, the region inside $\mathbf{c}_{out}$, and the region between them (figure 7.1). The region inside $\mathbf{c}_{in}$ is an *inflow sector*: all stream lines starting there end in $\mathbf{x}_0$ by forward integration. Similarly, the region inside $\mathbf{c}_{out}$ is an *outflow sector*. As in the 2D case, both inflow and outflow sectors are *parabolic sectors*. The region between $\mathbf{c}_{in}$ and $\mathbf{c}_{out}$ may be either a *hyperbolic sector* or an *elliptic sector*. In case of a



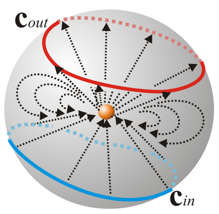
(a) Inflow surface described by a closed curve $\mathbf{c}_{in}$ on $\mathbf{s}$.



(b) Outflow surface described by a closed curve $\mathbf{c}_{out}$ on $\mathbf{s}$.



(c) Outflow sector (red area) inside $\mathbf{c}_{out}$, and inflow sector (blue area) inside $\mathbf{c}_{in}$.



(d) Elliptic sector between $\mathbf{c}_{in}$ and $\mathbf{c}_{out}$.

Figure 7.1: Segmentation of a 3D critical point by one inflow and one outflow curve. For modeling purposes certain restrictions have been imposed onto the separation curves: they have to be closed and need to be either inflow or outflow.

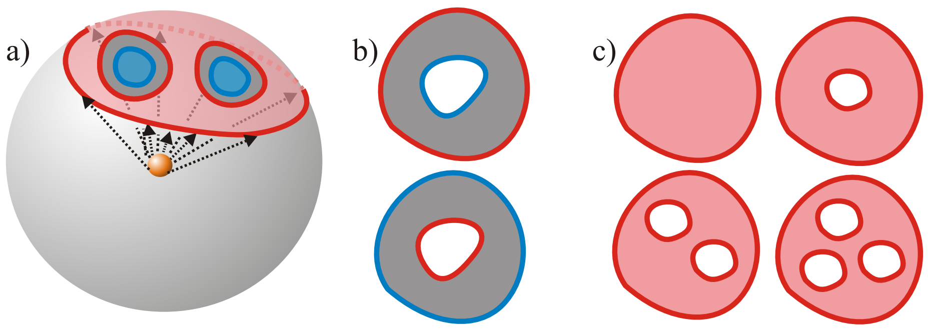
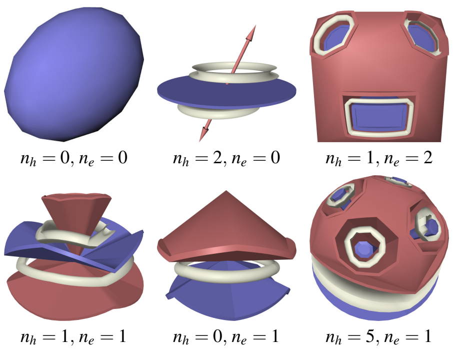


Figure 7.2: Segmentation of a 3D critical point by multiple inflow/outflow curves. a) an outflow sector is further divided into two hyperbolic sectors (gray) and two inflow sectors (blue); b) hyperbolic and elliptic sectors are topologically equivalent to a punctured disk; c) parabolic sectors are topologically equivalent to a disk with an arbitrary number of holes.

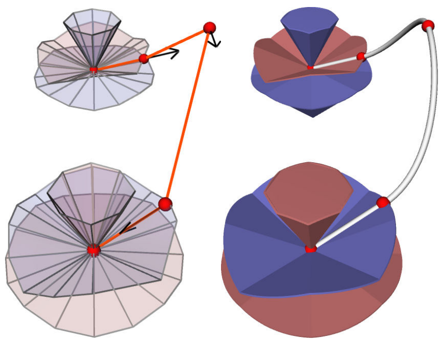
hyperbolic sector, all stream lines pass by $\mathbf{x}_0$, for an elliptic sector all stream line start and end in $\mathbf{x}_0$.

After showing how $\mathbf{s}$ is segmented by one inflow and one outflow curve, we consider the presence of additional inflow/outflow curves. Doing so, each of the sectors can be further subdivided into more sectors. Figure 7.2a shows an example where an outflow sector is divided into two hyperbolic and two inflow sectors. This example also shows a property about the shape of the different sectors: each hyperbolic and each elliptic sector have a strip shaped topology, i.e., they are topologically equivalent to a punctured disk (figure 7.2b). On the contrary, parabolic sectors are topologically equivalent to a disk with an arbitrary number of holes, or they cover the whole sphere (figure 7.2c).

The well-known first-order critical points fit into this system as well. They have a non-vanishing Jacobian and are therefore governed by the first-order partials. We can distinguish sources, sinks and saddles. Sources and sinks consist of one parabolic sector covering the whole surface of $\mathbf{s}$. A saddle consists of an outflow plane and two degenerate inflow surfaces (or the other way around, an inflow plane and two degenerate outflow surfaces). Hence, a saddle consists of two hyperbolic sectors. Figure 7.3a



(a) Modeled critical points with varying numbers n_h and n_e of hyperbolic and elliptic sectors.



(b) Modeled connector with one intermediate point. (left) Control polygon. (right) Visualization.

Figure 7.3: Modeled topological elements.

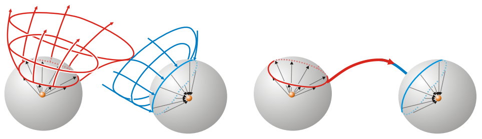


Figure 7.4: Limited control over the behavior of separation surfaces using connectors. (left) Intersecting separation surfaces. (right) Connector.

illustrates this with a sink and an attracting saddle.

In addition to our segmentation of a 3D critical point there are other topological features which we do not treat here. For example, consider a 2D critical point with imaginary eigenvalues of the Jacobian producing spiral-shaped stream lines around the point. Such stream lines may also occur in an inflow/outflow surface of a 3D critical point, which we did not include into the classification here.

The modeling of a critical point can now be done as follows: We place a number of control points $\mathbf{r}_i$ and $\mathbf{a}_j$ on the sphere $\mathbf{s}$ around $\mathbf{x}_0$ and provide them with vectors $\mathbf{v}_{\mathbf{r}_i}$ and $\mathbf{v}_{\mathbf{a}_j}$. These vectors describe the behavior of $\mathbf{v}$ around $\mathbf{x}_0$. In particular, an outflow surface is represented by a closed polygon $R = (\mathbf{r}_0, \ldots, \mathbf{r}_n)$ on $\mathbf{s}$ with the supplied vectors $\mathbf{v}_{\mathbf{r}_i} = \lambda_{\mathbf{r}_i}(\mathbf{r}_i - \mathbf{x}_0)$ for positive $\lambda_{\mathbf{r}_i}$. This way, only the length $\lambda_{\mathbf{r}_i}$ of the vectors $\mathbf{v}_{\mathbf{r}_i}$ can be chosen by the user. An inflow surface is modeled in a similar way as a closed polygon $A = (\mathbf{a}_0, \ldots, \mathbf{a}_m)$ with $\mathbf{v}_{\mathbf{a}_j} = \lambda_{\mathbf{a}_j}(\mathbf{x}_0 - \mathbf{a}_j)$ and $\lambda_{\mathbf{a}_j} > 0$.

Once all inflow and outflow surfaces are modeled, all regions between an inflow and an outflow surface have to be marked either as a hyperbolic or elliptic sector. The remaining areas are parabolic sectors and do not have to be specified.

The depiction of the modeled critical points follows the approach of section 2.4. Figure 7.3a shows a collection of critical points with varying number of hyperbolic and elliptic sectors.

Connectors and Stream Lines

For 3D vector fields, separatrices are certain stream surfaces separating areas of different flow behavior. The inflow/outflow surfaces of a critical point have exactly this behavior. Thus, separation surfaces can be obtained by a stream surface integration starting from a critical point into the direction of an inflow/outflow surface. Unfortunately, stream surfaces tend to have a rather complicated behavior making modeling these surfaces a cumbersome task. Therefore, we adapt the concept of connectors for our purposes by modeling not the separation surfaces but only their intersection curves. These connectors are certain stream lines starting in the inflow surface of a critical point and ending in the outflow surface of another (or the same) critical point. Using them for modeling the topological skeleton gives a certain control over the behavior of the separation surfaces, though it is not a full replacement. Figure 7.4 illustrates this.

In order to model the desired connectors, we first have to define their start and end points. The start point is specified by a critical point $\mathbf{x}_0$, an outflow surface described by the closed polygon $R = (\mathbf{r}_0, ..., \mathbf{r}_n)$, and a point on this polygon. Similarly, the end point of a connector is defined by a point on an inflow surface. In addition, we allow a number of intermediate points which are supplied with vectors; the connector is supposed to interpolate them both in location and tangent direction. Figure 7.3b shows a connector between two critical points which is defined by starting point, end point, and one intermediate point. To visualize a connector, we assume a piecewise C^1 continuous cubic curve interpolating all points both in location and tangent direction.

A vector field may have areas where only a few (or even no) topological features are present. For these areas the modeling of the topological skeleton does not give any information to build the vector field. To deal with such areas, we additionally allow the user to place arbitrary stream lines into the domain. They are modeled - similar to the connectors - as a sequence of points supplied with a vector information. Then the stream lines are the interpolating (both in location and tangent) piecewise C^1 continuous cubic curves. These stream lines are later incorporated into the construction of the vector field, even though they are not part of the topological skeleton.

7.1.3 Construction of the Vector Field

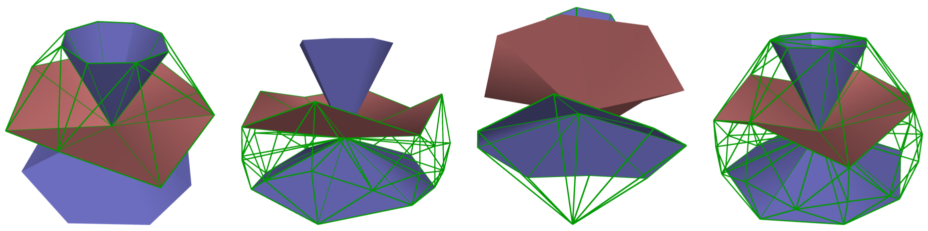
Once the topological skeleton is modeled, we have to construct a vector field which has the specified skeleton. We use a piecewise linear vector field, i.e. we construct a tetrahedrization of the domain where each of its vertices is supplied with a vector. The construction is done in a fully-automatic way.

Critical Points

To construct the tetrahedrization around a critical point $\mathbf{x}_0$, we extend the approach of [TSH00] and [The02] to 3D: a vertex with a zero vector is placed at $\mathbf{x}_0$, around this a number of tetrahedra sharing $\mathbf{x}_0$ as a vertex are placed. For these tetrahedra, we essentially use the vertices of the polygons defining inflow and outflow surfaces. We give the following algorithm to tetrahedrize the region around a critical point:

1. Consider a sphere $\mathbf{s}$ around $\mathbf{x}_0$ such that $\mathbf{s}$ does not intersect any sphere around another critical point. We assume that k outflow surfaces are present which are described by the k closed polygons $R_0 = (\mathbf{r}_{0,0}, ..., \mathbf{r}_{n_0,0}), ..., R_{k-1} = (\mathbf{r}_{0,k-1}, ..., \mathbf{r}_{n_{k-1},k-1})$. We further assume that ℓ inflow surfaces are described by the closed polygons $A_0 = (\mathbf{a}_{0,0}, ..., \mathbf{a}_{m_0,0}), ..., A_{\ell-1} = (\mathbf{a}_{0,\ell-1}, ..., \mathbf{a}_{m_{\ell-1},\ell-1})$.

2. Insert all start and end points of connectors into the closed polygons. If a connector starts at $\mathbf{x}_0$ in the i-th outflow surface, its starting point has to be inserted as a new vertex into R_i. Similarly, if a connector ends in $\mathbf{x}_0$ in the j-th inflow surface, its end point is inserted into A_j (cf. figure 7.3b). This is necessary to get consistent tetrahedrizations of critical points and connectors later on.



(a) Hyperbolic sector. (b) Elliptic sector: an auxiliary closed polygon was inserted. (c) Both parabolic sectors. (d) Complete tetrahedrization.

Figure 7.5: Tetrahedrization of different sectors of a critical point.

3. Tetrahedrize all hyperbolic sectors. If the modeler gives a hyperbolic sector between the i-th outflow surface and the j-th inflow surface, we triangulate the strip between R_i and A_j by using a constrained Delaunay triangulation. The obtained triangles are converted to tetrahedra by connecting them with $\mathbf{x}_0$ (figure 7.5a).

4. Tetrahedrize all elliptic sectors. If there is an elliptic sector between the i-th outflow surface and the j-th inflow surface, we have to construct an auxiliary closed polygon $H_{i,j} = (\mathbf{h}_{0,i,j}, ..., \mathbf{h}_{n_{i,j},i,j})$ between R_i and A_j. To do so, we imagine a constrained Delaunay triangulation between R_i and A_j. If the edge $(\mathbf{r}_{e,i}, \mathbf{a}_{f,j})$ is part of this triangulation, we insert the point $(\mathbf{r}_{e,i} + \mathbf{a}_{f,j})/2$ into $H_{i,j}$ and provide it with the vector $\mathbf{a}_{f,j} - \mathbf{r}_{e,i}$. Then we apply a constrained Delaunay triangulation between R_i and $H_{i,j}$, and another one between $H_{i,j}$ and A_j. The obtained triangles are tetrahedrized by connecting them with $\mathbf{x}_0$ (figure 7.5b).

5. Tetrahedrize all parabolic sectors. All remaining areas on $\mathbf{s}$ are triangulated by a constrained Delaunay triangulation. If necessary, additional inflow/outflow points on $\mathbf{s}$ have to be inserted. The triangles are tetrahedrized by connecting them with $\mathbf{x}_0$ (figure 7.5c).

Connectors and Stream Lines

To tetrahedrize connectors and stream lines, we keep in mind that we modeled them as piecewise cubic curves. The stream lines of a linear vector field are certain exponential curves which can be expressed in a closed form [Nie97]. Fortunately, it turns out that the class of non-planar cubic curves is contained in the class of stream lines of linear vector fields. We give the following

Theorem 2 *Given are 4 non-coplanar points $\mathbf{p}_0, \dots, \mathbf{p}_3$ which are equipped with the 3D vectors $\mathbf{v}_0, \dots, \mathbf{v}_3$. This way a linear vector field $\mathbf{v}$ is defined inside the tetrahedron $\mathbf{p}_0, \dots, \mathbf{p}_3$. If $\mathbf{v}_0, \dots, \mathbf{v}_3$ are chosen as*

$$\mathbf{v}_0 = \lambda_0 (\mathbf{p}_1 - \mathbf{p}_0) \quad , \quad \mathbf{v}_3 = \lambda_3 (\mathbf{p}_3 - \mathbf{p}_2) \tag{7.1}$$

$$\mathbf{v}_1 = \frac{1}{3}\lambda_3 (\mathbf{p}_1 - \mathbf{p}_0) + \frac{2}{3}\lambda_0 (\mathbf{p}_2 - \mathbf{p}_1) \tag{7.2}$$

$$\mathbf{v}_2 = \frac{2}{3}\lambda_3 (\mathbf{p}_2 - \mathbf{p}_1) + \frac{1}{3}\lambda_0 (\mathbf{p}_3 - \mathbf{p}_2) \tag{7.3}$$

for certain $\lambda_0, \lambda_3 > 0$, then the following property holds: the stream line of $\mathbf{v}$ starting in $\mathbf{p}_0$ is identical to the cubic Bézier curve defined by the Bézier points $\mathbf{p}_0, \dots, \mathbf{p}_3$.

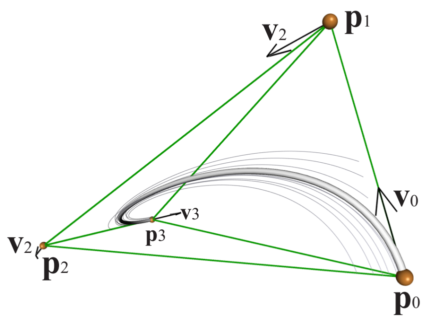
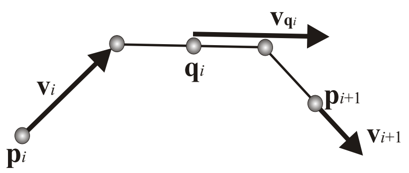
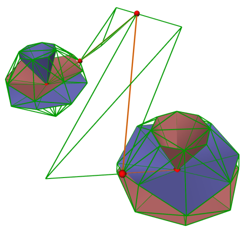


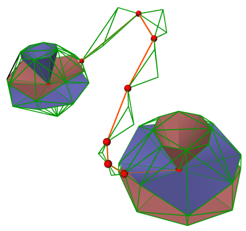
Figure 7.6: Setup of theorem 2: one of the stream lines of the linear vector field coincides with the cubic Bézier curve.



(a) Dividing an almost planar cubic part into two quadratic parts.



(b) Intersecting tetrahedra.



(c) After repeated subdivision of intersecting tetrahedra.

Figure 7.7: Subdivision of a connector or stream line.

Figure 7.6 illustrates the setup of theorem 2. To prove this theorem, we consider the Bézier curve $\mathbf{x}(t) = \sum_{i=0}^{3} B_i^3(t)\,\mathbf{p}_i$ where $B_i^3(t)$ are the cubic Bernstein polynomials. We have to show that $\dot{\mathbf{x}}(t) \times \mathbf{v}(\mathbf{x}(t)) = (0,0,0)^T$ for any t. Inserting (7.1)-(7.3) into this equation, this is a straightforward exercise in algebra. Note that the vector field generated by (7.1)-(7.3) generally has only one cubic curve as a stream line: the stream lines around $\mathbf{x}(t)$ are generally no polynomial curves.

Theorem 2 does not only show that non-planar cubic curves are contained in piecewise linear vector fields, it also shows how to construct this vector field. Given a cubic curve by its Bézier polygon $\mathbf{p}_0,\dots,\mathbf{p}_3$, we add the tetrahedron $\mathbf{p}_0,\dots,\mathbf{p}_3$ with the vectors $\mathbf{v}_0,\dots,\mathbf{v}_3$ obtained from (7.1)-(7.3) to the tetrahedrization. Unfortunately, this approach is not applicable if $\mathbf{p}_0,\dots,\mathbf{p}_3$ are nearly coplanar, which leads to a degenerate tetrahedron.[1] In this case we subdivide the cubic curve into two quadratic pieces (figure 7.7a): if the cubic between the points-with-vectors $(\mathbf{p}_i,\mathbf{v}_i)$ and $(\mathbf{p}_{i+1},\mathbf{v}_{i+1})$ is nearly planar, we insert a new point-with-vector $(\mathbf{q}_i,\mathbf{v}_{\mathbf{q}_i})$ with $\mathbf{q}_i = (\mathbf{p}_i+\mathbf{v}_i+\mathbf{p}_{i+1}-\mathbf{v}_{i+1})/2$ and $\mathbf{v}_{\mathbf{q}_i} = (\mathbf{p}_{i+1}-\mathbf{v}_{i+1})-(\mathbf{p}_i+\mathbf{v}_i)$. Then $(\mathbf{p}_i,\mathbf{v}_i)$ and $(\mathbf{q}_i,\mathbf{v}_{\mathbf{q}_i})$ are coplanar, as well as $(\mathbf{q}_i,\mathbf{v}_{\mathbf{q}_i})$ and $(\mathbf{p}_{i+1},\mathbf{v}_{i+1})$. We then construct a piecewise linear vector field yielding the parabola between these points as described in [The02].

Once we have constructed one (or two) tetrahedra for each cubic segment of each connector and stream line, these tetrahedra might intersect each other as well as other tetrahedra coming from critical points. In this case, they have to be subdivided into two new tetrahedra by subdividing the underlying cubic or quadratic Bézier curve. This process is repeated until no tetrahedra intersect any more (figures 7.7b–c).

After tetrahedrizing all critical points and connectors (i.e., the topological skeleton), we have to fill the remaining areas of the domain which are not tetrahedrized yet.

[1]In fact, planar cubics are *not* contained in the class of stream lines of linear vector fields.

To do so, we define a bounding box around the domain and apply a constrained Delaunay tetrahedrization incorporating the 8 vertices of the bounding box. For this task, we used an adaption of the freely available library TetGen [Si02]. This way a number of new vertices may be created. To assign them with vectors (as well as to assign the vertices of the bounding box with vectors), we compute a weighted average of the vectors of all vertices sharing an edge with the new vertex; the weights are the inverse of the distance of the vertices.

Topological Completeness and Consistency

Our vector field construction approach described above guarantees that all topological features (critical points and connectors) of the skeleton are contained in the vector field. However, we can *not* guarantee that no additional critical points and connectors appear.

The appearance of additional connectors is due to the fact, that our modeling approach does not allow full control over the behavior of the separation surfaces. To achieve this one would need to model them as e.g. parametric surfaces, which is a cumbersome or even impossible task. Furthermore, there seems to be no approach to create a vector field with stream surfaces coinciding to such parametric surfaces. However, depending on the scenario one can try to model stream lines in order to give the field (and its stream surfaces) a certain direction in a specific area (cf. figure 7.10b).

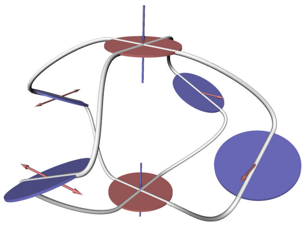
There are two main reasons for the appearance of additional critical points. First, the indices of the modeled critical points may sum up to a high (positive or negative) number. This happens if e.g. only sources have been modeled, but no sinks or saddles. As vector fields usually tend to have an index close to zero, additional critical points are likely to appear in this case. Second, additional critical points may be due to an inappropriate tetrahedrization of the remaining areas or an inappropriate choice of the vectors at the vertices of the bounding box. A more involved strategy may make them disappear. However, such an optimized tetrahedrization or choice of vectors is beyond the scope of this thesis.

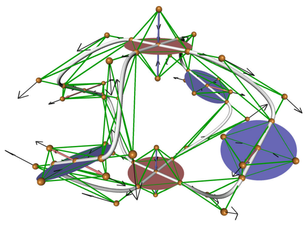
7.1.4 Examples

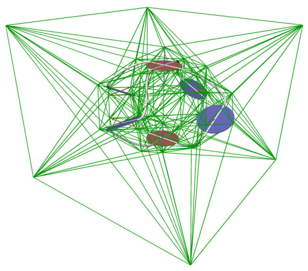
In this section we demonstrate our approach to model a number of 3D vector fields of different topological behavior.

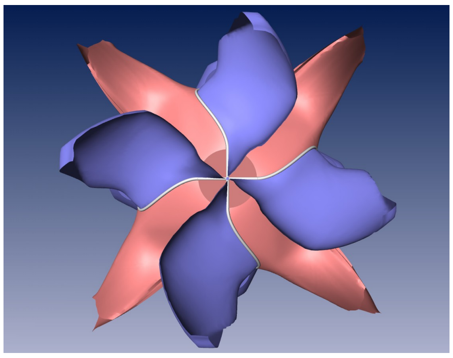
Figure 7.8a shows a modeled topological skeleton consisting of 6 critical points and 8 connectors. Each of the critical points consists of two hyperbolic sectors and is actually a first order saddle point. Each of the connectors was defined by specifying start and end point and omitting any intermediate points. Thus, each connector consists of one cubic segment. Figure 7.8b shows the result of the tetrahedrization of the critical points and the connectors. In this figure we can clearly see that each connector is constructed in one tetrahedron. Figure 7.8c shows the complete tetrahedrization of the piecewise linear vector field consisting of 256 tetrahedra. Figures 7.8d and 7.8e show different visualizations of the newly constructed vector field. Figure 7.8d shows a stream surface integration of the separation surfaces. They are color coded in red (outflow surface) and blue (inflow surface). Figure 7.8e shows the extraction of saddle connectors [TWHS03] revealing that they coincide with the modeled connectors of figure 7.8a. In addition, figure 7.8e shows a number of illuminated stream lines [ZSH96].

Figure 7.9 shows another modeled topological skeleton. We included this example to demonstrate that our approach can also handle a higher number of connectors


(a) Modeled topological skeleton.


(b) Tetrahedrization of critical points and connectors.


(c) Complete tetrahedrization.


(d) Separation surfaces of constructed vector field. View from top.

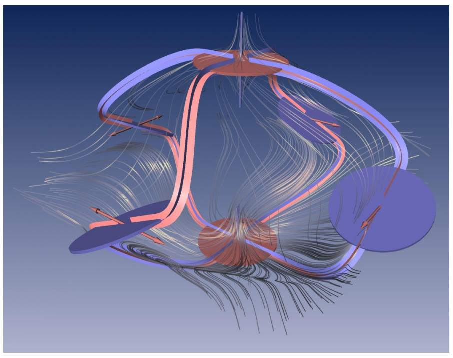
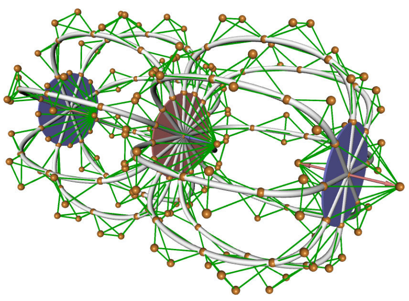

(e) Saddle connectors and stream lines of constructed vector field.

Figure 7.8: Example 1.


(a) Tetrahedrization of the modeled skeleton.

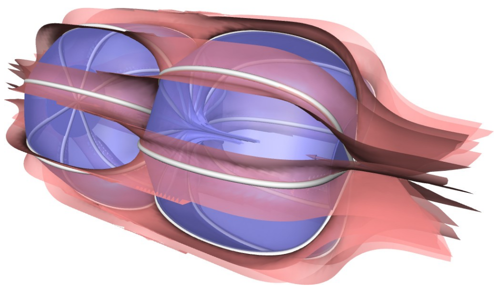
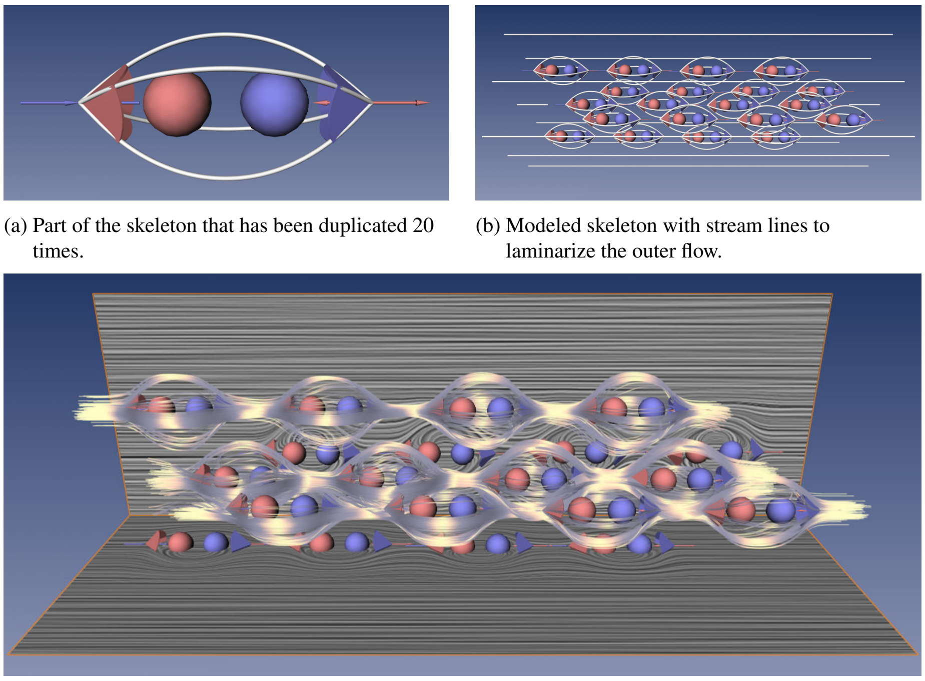

(b) Separation surfaces and modeled connectors.

Figure 7.9: Example 2.


(a) Part of the skeleton that has been duplicated 20 times.

(b) Modeled skeleton with stream lines to laminarize the outer flow.

(c) Constructed vector field visualized using illuminated stream lines and LIC.

Figure 7.10: Example 3.

between two critical points. In this example, we alternately connected the outflow surface of the critical point in the middle with the inflow surfaces of the two other critical points. All the connectors are modeled without intermediate points. Figure 7.9a illustrates the tetrahedrization of the skeleton, showing that each of the connectors is represented by 4 quadratic pieces. The automatic subdivision (which is responsible for this) first subdivided the modeled planar cubic curves into two quadratic curves, then each of them was subdivided one more time to remove the intersections of the defining tetrahedra of adjacent connectors. Figure 7.9b shows the integrated separation surfaces in a semi-transparent way. Note how the red separation surface follows all connectors to the other two critical points. This example exemplifies that the control of separation surfaces using connectors is possible to a certain degree.

In figure 7.10 we modeled a vector field consisting of 80 critical points, 80 connectors, and 14 additional stream lines. It describes a laminar flow in which 20 regions of different flow behavior are inserted. Each of these regions consists of a source and a sink and is bounded by the separation surfaces of two critical points with two hyperbolic sectors (figure 7.10a). We modeled 4 connectors between each of these critical points. The regular setup of this example ensures that the two separation surfaces coincide. Inside the areas of these surfaces we have a simple behavior: every stream line starts in the source and ends in the sink. Outside these areas we modeled a turbulence-free flow around it, which is laminar in a certain distance from the inserted regions. To do so, we included a number of straight stream lines into the flow. Figure 7.10c shows a visualization of the resulting piecewise linear vector field consisting of 8198 tetrahedra: two inserted LIC planes reveal the laminar flow in a distance of the inserted areas as well as the flow behavior inside it. In addition, a number of illuminated stream lines are inserted.

7.2 Vortex Core Lines

Flow fields play a vital role in many research areas. Examples are burning chambers, turbomachinery and aircraft design in industry as well as visualization and control of blood flow in medicine. As the resolution of numerical simulations and experimental measurements like PIV have evolved significantly in the last years, the challenge of understanding the intricate flow structures within massive result data sets has made automatic feature extraction necessary.

Among the features of interest are vortical structures. Vortices play a major role due to their wanted or unwanted effects on the flow. In turbomachinery design, vortices reduce efficiency, whereas in burning chambers, vortices have to be controlled to achieve optimal mixing of oxygen and fuel. In aircraft design, vortices can both increase and decrease lift.

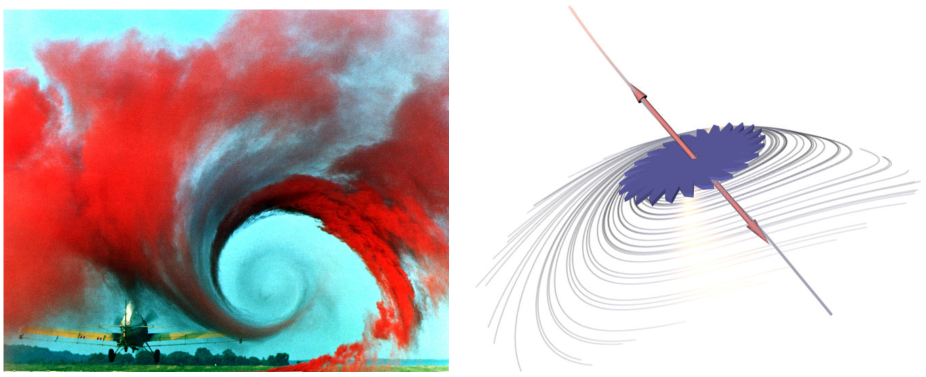
In this section we show how vortices can be assessed using the same methods that we use for extracting topological features, i.e., the Unified Feature Extraction Architecture. We link the areas of topology and vortex analysis not only by means of the extraction methods, but also by showing how the core of a vortex can be defined as a separatrix of a topological skeleton of a derived vector field. This opens opportunities for future work, where topological tools, like e.g. simplification, can be applied to the field of vortex analysis.

While [PVH$^+$02] and [PR99] give a thorough overview of algorithms for the treatment of vortical structures, we give a short introduction here. They can be classified in two major categories:

- *Vortex region detection* is based on scalar quantities that are used to define a vortex as a spatial region where the quantity exhibits a certain value range. We refer to them as *vortex region quantities*. Examples of this are regions of high vorticity magnitude, negative λ_2-criterion [JH95], or positive Q-criterion [Hun87] (see also section 2.1.3). Isosurfaces or volume rendering are common approaches for visualizing these quantities, which requires the choice of thresholds and appropriate isovalues or transfer functions. As shown in [RP96], this can become a difficult task for some settings.

- *Vortex core line extraction* aims at extracting line type features that are regarded as centers of vortices. Different approaches exist, e.g. cores of swirling motion [SH95b, RP98, PR99, RSVP02, JMT02, WSTH07], vorticity lines corrected towards pressure minima [BS95] or λ_2 minima [SRE05], or lines of extremal scalar value [MK97, SWH05b, SWTH07]. The extraction of such lines is parameter free in the sense that their definition does not refer to a range of values. This eliminates the need of choosing certain thresholds.

In the following we describe three different approaches to extracting vortex core lines, which have been developed or extended in the course of this thesis as applications of the Unified Feature Extraction Architecture. First, we aim at the extraction of vortex core lines as centers of swirling motion, in particular we developed a novel mathematical description of swirling motion for path lines (section 7.2.1). The other two methods aim at the extraction of Galilean invariant vortex core lines, i.e., the extracted features remain unchanged when a constant vector is added to the flow field.



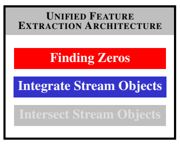
(a) Wake vortex study from NASA Langley Research Center [NAS90]. The unsteady flow around a starting agricultural plane is made visible using smoke injection. A huge pattern of *swirling particle motion* is created by the aircraft's wake vortex.

(b) Flow pattern around a focus saddle. Swirling motion of *stream lines* takes place in the blue plane around the real eigenvector denoted by the red arrows. This flow pattern inspired the definition of swirling motion by Sujudi/Haimes [SH95b].

Figure 7.11: Swirling motion in steady and unsteady flows.

This is done by considering ridge or valley lines of Galilean invariant vortex region quantities. In section 7.2.2 we show how to extract ridges and valleys based on second order derivatives. In section 7.2.3 we show how to extract similar structures based on a topological analysis and using first order derivatives only.

7.2.1 Cores of Swirling Motion



One way to assess vortices in experiments is to emit particles into the flow and to examine their behavior: patterns of swirling flow indicate vortices. This has been done in the experiment shown in figure 7.11a. By injecting smoke, i.e., a huge amount of particles, swirling flow caused by the wake vortex becomes visible. For numerical and measured data sets, Sujudi and Haimes [SH95b] proposed a scheme to extract centers of swirling flow. Peikert et al. formulated the idea of Sujudi/Haimes using the Parallel Vectors operator and presented a fast and robust extraction technique [PR99]. These methods assess the behavior of *stream lines* only.

However, most flow phenomena are unsteady in nature. In unsteady flows (as shown in figure 7.11a), particle motion is described by *path lines* instead of stream lines. This generally gives different swirling patterns. We aim at extracting the cores of swirling particle motion in unsteady flows based on the behavior of path lines. To do so, we develop a novel mathematical characterization of such cores as a generalization of the original idea of Sujudi/Haimes. We do this for 2D and 3D flows. In the latter case, the resulting core structures are lines sweeping over time, i.e., surfaces in the space-time domain. At a single time step, particles group around these core lines forming patterns of swirling motion similar to figure 7.11a. That is why we refer to those features as *swirling particle cores*. Mathematically, they are characterized by the coplanarity of three 4D vectors. In order to extract them, we show how to re-formulate the problem using the Parallel Vectors operator [PR99].

All cores of swirling motion can be extracted using the Unified Feature Extraction

Architecture by applying the algorithms for Finding Zeros and Integrating Stream Objects. In most cases we can apply extraction schemes known from previous chapters. However, the case of 3D unsteady flows with core lines sweeping over time has not been covered yet. Bauer et al. [BP02] proposed a marching-cubes-like algorithm to track these centers over time. At the end of this section we introduce a new method for tracking these cores using the concept of Feature Flow Fields.

First, we will discuss the different approaches to swirling motion. This yields an unified notation of swirling motion in 2D and 3D flows. Afterwards, we discuss the extraction of swirling motion cores.

Swirling Motion

The term "swirling motion" refers to the behavior of characteristic curves of the examined field. Hence, one may find swirling motion of

- stream lines in steady fields,
- stream lines in unsteady fields,
- path lines in unsteady fields.

All three cases can be found in 2D as well as 3D fields, summing up to a total of six cases. Patterns of spiraling *stream lines* in 2D and 3D flows have already been treated in the literature. These patterns are assessed by examining eigenvalues and eigenvectors of the Jacobian matrix $\mathbf{J}$ of the respective flow field $\mathbf{v}$. A necessary condition for spiraling stream lines in $\mathbf{v}$ is that $\mathbf{J}$ has a pair of conjugate complex eigenvalues.

The Jacobian of a steady 2D flow field has either two real or one pair of conjugate complex eigenvalues. Swirling motion occurs in the latter case only – stream lines spiraling around a common point. The velocity at this point must be zero, i.e., $\mathbf{v}(x,y) = \mathbf{0}$. This means that cores of swirling motion in 2D steady flow fields are certain types of critical points, namely foci and centers. Thus, they can be treated using steady flow field topology as described in section 2.2.1.

Following (2.14), the stream lines of an unsteady 2D flow field always stay in the same given time slice t_0. Thus, swirling motion in a single time slice can be captured by applying the scheme known from the steady case. By changing the given time slice, the critical points will move over time and form line-type structures in space-time. In other words, this can be treated using a topological analysis of the underlying one-parameter-dependent vector field as discussed in section 2.3.1.

Centers of swirling motion in steady 3D flows have first been treated by Sujudi and Haimes [SH95b]. Their inspiration was the flow pattern around a certain type of critical point: a focus saddle (see figure 7.11b for an illustration). Here, the Jacobian of the flow field has one real and two complex eigenvalues. The eigenvectors corresponding to the complex eigenvalues span a plane in which the flow spirals around the critical point. The eigenvector corresponding to the real eigenvalue denotes the axis of rotation. Sujudi and Haimes generalized this flow pattern to non-critical points by considering the so-called reduced velocity. At a point $\mathbf{x}$, the reduced velocity $\mathbf{w}(\mathbf{x})$ is given as the projection of the steady flow field $\mathbf{v}(\mathbf{x})$ to the plane normal to the real eigenvector $\mathbf{e}$ by

$$\mathbf{w}(\mathbf{x}) = \mathbf{v}(\mathbf{x}) - (\mathbf{v}(\mathbf{x}) \cdot \mathbf{e}(\mathbf{x}))\,\mathbf{e}(\mathbf{x}). \tag{7.4}$$

They show that centers of swirling flow are line-type structures where $\mathbf{w}(\mathbf{x}) = 0$. Peikert et al. [PR99] formulated this using the Parallel Vectors operator and showed that $\mathbf{w}(\mathbf{x}) = 0$ is equivalent to $\mathbf{v}(\mathbf{x})||\mathbf{e}(\mathbf{x})$, i.e., $\mathbf{v}$ and $\mathbf{e}$ are parallel.

At each time step of an unsteady 3D flow field, swirling stream line behavior can be assessed using the scheme known from the steady case. This yields lines sweeping over time, resulting in 4D surfaces in the space-time domain.

To the best of our knowledge, patterns of spiraling *path lines* in 2D and 3D unsteady flows have not been treated in the literature prior to this thesis. In the following we develop a mathematical characterization of swirling particle cores as a generalization of the original idea of Sujudi/Haimes. We do this for 3D unsteady flows and refer the reader to [WSTH07] for the 2D case.

We aim at extracting swirling particle cores of an unsteady 3D flow field $\mathbf{v}(x,y,z,t)$, i.e., locations around which spiraling patterns of path lines occur. Following (2.13), path lines of $\mathbf{v}$ are stream lines of the steady 4D vector field

$$\mathbf{p}(x,y,z,t) = \begin{pmatrix} \mathbf{v}(x,y,z,t) \\ 1 \end{pmatrix} = \begin{pmatrix} u(x,y,z,t) \\ v(x,y,z,t) \\ w(x,y,z,t) \\ 1 \end{pmatrix}. \tag{7.5}$$

There is no existing tool to identify swirling motion in a steady 4D vector field. In the following we develop a new approach. The key is, again, the eigensystem of the Jacobian of $\mathbf{p}$. In the 3D unsteady setting, the Jacobian of $\mathbf{p}$ is

$$\mathbf{J}(\mathbf{p}) = \begin{bmatrix} u_x & u_y & u_z & u_t \\ v_x & v_y & v_z & v_t \\ w_x & w_y & w_z & w_t \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.6}$$

and has the eigenvalues $e_1, e_2, e_3, 0$ with the respective four eigenvectors

$$\begin{pmatrix} \mathbf{e}_1 \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{e}_2 \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{e}_3 \\ 0 \end{pmatrix} =: \mathbf{e}^s, \mathbf{f}, \tag{7.7}$$

where $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ are the eigenvectors of the spatial Jacobian

$$\mathbf{J}_s(\mathbf{v}) = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{bmatrix} \tag{7.8}$$

and the fourth eigenvector $\mathbf{f}$ can be written as

$$\mathbf{f}(x,y,z,t) = \begin{pmatrix} +\det(\mathbf{v}_y, \mathbf{v}_z, \mathbf{v}_t) \\ -\det(\mathbf{v}_z, \mathbf{v}_t, \mathbf{v}_x) \\ +\det(\mathbf{v}_t, \mathbf{v}_x, \mathbf{v}_y) \\ -\det(\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z) \end{pmatrix}. \tag{7.9}$$

Note, that $\mathbf{f}$ is the FFF for tracking critical points in 3D unsteady vector fields (section 5.1.1) and the eigenvalue corresponding to $\mathbf{f}$ is always zero.[2] Therefore, $\mathbf{J}(\mathbf{p})$ has always one real eigenvalue and only the following cases can occur:

[2]This can be explained as follows: in order to track critical points of $\mathbf{v}$, the feature flow field $\mathbf{f}$ was designed ([TS03, WTHS07] and section 5.1.1) such that the values of $\mathbf{v}$ do not change along the stream lines of $\mathbf{f}$. In other words, the directional derivative of $\mathbf{v}$ in direction of $\mathbf{f}$ is zero. This means that $\mathbf{J}(\mathbf{v}) \cdot \mathbf{f} = \mathbf{0}$ and consequently $\mathbf{J}(\mathbf{p}) \cdot \mathbf{f} = 0 \cdot \mathbf{f}$. Hence, $\mathbf{f}$ necessarily is an eigenvector of $\mathbf{J}(\mathbf{p})$ corresponding to the eigenvalue 0.

- All eigenvalues of $\mathbf{J}(\mathbf{p})$ are real.
- $\mathbf{J}(\mathbf{p})$ has a pair of conjugate complex eigenvalues and two real eigenvalues – let them be sorted such that e_1, e_2 are complex and e_3 is real.

Since complex eigenvalues are a necessary condition, swirling motion is only possible in the latter case. At any given point $\mathbf{x}$ in the 4D domain, the eigenvectors corresponding to e_1, e_2 span a plane P_c in which locally the swirling motion occurs. The two real eigenvectors $\mathbf{e}^s$ and $\mathbf{f}$ denote the part of the flow which is independent of swirling – they span a plane P_r in which no swirling occurs at all. In order to see what the core of swirling motion in 4D is, consider the following rephrasing of the definitions of swirling motion cores in other dimensions:

> *Although a point* $\mathbf{x}$ *on the core structure is surrounded by spiraling integral curves, the flow vector at* $\mathbf{x}$ *itself is solely governed by the non-swirling part of the flow.*

This is a direct generalization of the "reduced velocity"-idea of Sujudi/Haimes. For our case this means that $\mathbf{x}$ is a point on the swirling particle core if the flow vector $\mathbf{p}(\mathbf{x})$ lies in the plane of non-swirling flow P_r, i.e., the plane spanned by $\mathbf{e}^s$ and $\mathbf{f}$. In other words, the swirling particle cores are at locations where

$$\lambda_1 \mathbf{p} + \lambda_2 \mathbf{e}^s + \lambda_3 \mathbf{f} = \mathbf{0} \quad \text{with} \quad \lambda_1^2 + \lambda_2^2 + \lambda_3^2 > 0. \tag{7.10}$$

This is a coplanarity problem: swirling particle cores are at locations where the 4D vectors $\mathbf{p}$, $\mathbf{e}^s$ and $\mathbf{f}$ are coplanar. We call the operator solving this equation the *Coplanar Vectors* operator, which reads in the general setting

$$\lambda_1 \mathbf{a} + \lambda_2 \mathbf{b} + \lambda_3 \mathbf{c} = \mathbf{0} \quad \text{with} \quad \lambda_1^2 + \lambda_2^2 + \lambda_3^2 > 0. \tag{7.11}$$

In order to show that our approach is reasonable, we study what happens if we require the flow field $\mathbf{v}$ to be steady, i.e., $\mathbf{v}(x,y,z,t) = \mathbf{v}(x,y,z,t_0)$. In this setting, path lines coincide with stream lines and our approach needs to reduce to the steady case, i.e., the method of Sujudi/Haimes. As the temporal derivative $\mathbf{v}_t = 0$, the fourth eigenvector becomes $\mathbf{f} = (0,0,0,-\det(\mathbf{v}_x,\mathbf{v}_y,\mathbf{v}_z))^T$ following (7.9), and the coplanarity condition (7.10) reads

$$\lambda_1 \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix} + \lambda_2 \begin{pmatrix} \mathbf{e}_3 \\ 0 \end{pmatrix} + \lambda_3 \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\det(\mathbf{v}_x,\mathbf{v}_y,\mathbf{v}_z) \end{pmatrix} = \mathbf{0}. \tag{7.12}$$

The last component of this equation requires $\lambda_1 = \lambda_3 \det(\mathbf{v}_x,\mathbf{v}_y,\mathbf{v}_z)$. Hence, in the steady setting our approach reduces to $\mathbf{v}||\mathbf{e}_3$, i.e., the method of Sujudi/Haimes.

In the following we show how all six above discussed cases of swirling motion can be written using a unified notation. Let $\mathbf{u}$ be the autonomous system of the characteristic curves in question, i.e., $\mathbf{v}$ for the steady case (2.9), $\mathbf{s}$ for the stream lines of an unsteady flow (2.14), and $\mathbf{p}$ for path lines (2.13). Let $\mathbf{e}_i$ be the eigenvectors corresponding to the real eigenvalues of $\mathbf{J}(\mathbf{u})$. The point $\mathbf{x}$ is part of the respective core of swirling motion, if $\mathbf{u}(\mathbf{x})$ lies in the span of $\mathbf{e}_i(\mathbf{x})$. In other words, this reads

$$\lambda_1 \mathbf{u}(\mathbf{x}) + \sum \lambda_i \mathbf{e}_i(\mathbf{x}) = \mathbf{0} \quad \text{with} \quad \sum \lambda_i^2 > 0. \tag{7.13}$$

Depending on the dimension of $\mathbf{u}$, this is equivalent to

	2D	3D
steady stream lines	CRITICAL POINTS $\lambda\mathbf{v} = \mathbf{0}$ *CP finder* can be treated using [HH89]	SWIRLING STREAM LINE CORES $\lambda_1\mathbf{v} + \lambda_2\mathbf{e} = \mathbf{0}$ *PV operator* original idea of Sujudi/Haimes treated in [SH95b, PR99]
unsteady stream lines	TRACKED CRITICAL POINTS $\lambda_1\mathbf{s} + \lambda_2\mathbf{e} = \mathbf{0}$ *PV operator* can be treated using [TS03] and using [SH95b, PR99]	TRACKED STREAM LINE CORES $\lambda_1\mathbf{s} + \lambda_2\mathbf{e}^s + \lambda_3\mathbf{f} = \mathbf{0}$ *CV operator* treated in this thesis and [BP02]
unsteady path lines	SWIRLING PARTICLE CORES $\lambda_1\mathbf{p} + \lambda_2\mathbf{e} = \mathbf{0}$ *PV operator* treated in this thesis can be extracted using [SH95b, PR99]	SWIRLING PARTICLE CORES $\lambda_1\mathbf{p} + \lambda_2\mathbf{e}^s + \lambda_3\mathbf{f} = \mathbf{0}$ *CV operator* treated in this thesis

Table 7.1: Summary of swirling motion in 2D and 3D flows. Depending on the dimension of the autonomous system the conditions can be written using the notations of Critical Points (CP), Parallel Vectors (PV), and Coplanar Vectors (CV).

- extraction of critical points in 2D,
- solving the Parallel Vectors operator in 3D,
- solving the Coplanar Vectors operator in 4D.

Table 7.1 illustrates this. An interesting result of this is that critical points of 2D unsteady flows can be tracked using the Parallel Vectors approach. See [WSTH07] for more details on this.

Extraction

As already indicated in table 7.1, the extraction of swirling motion cores can be done by extracting certain critical points (2D steady case, see section 4.1), tracking them over time (2D unsteady stream line case, see section 5.1.1), or by solving the PV or CV operator. As shown in section 3.1.2, the PV operator can be solved by applying a FFF-based approach. It remains to show, how to extract structures defined by the CV operator. To do so, we show first that at least these specific CV problems can be treated as regular PV problems in a single time step and based on that, how to track the resulting line structures over time.

Obviously, swirling stream line cores of an unsteady 3D flow fulfill this condition since they are defined as an application of the PV operator to each single time step.

This is less obvious for swirling particle cores. We identified cores of swirling particle motion in unsteady 3D flows as locations where the three 4D vector fields

$\mathbf{p}, \mathbf{e}^s, \mathbf{f}$ are coplanar. This is given by (7.10), which reads component-wise

$$\lambda_1 \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} + \lambda_2 \begin{pmatrix} \mathrm{e}_1^s \\ \mathrm{e}_2^s \\ \mathrm{e}_3^s \\ 0 \end{pmatrix} + \lambda_3 \begin{pmatrix} \mathrm{f}_1 \\ \mathrm{f}_2 \\ \mathrm{f}_3 \\ \mathrm{f}_4 \end{pmatrix} = \mathbf{0}, \tag{7.14}$$

By setting $\lambda_1 = -\lambda_3 \mathrm{f}_4$ we can eliminate the fourth component, and the reformulation reads

$$\lambda_2 \underbrace{\begin{pmatrix} \mathrm{e}_1^s \\ \mathrm{e}_2^s \\ \mathrm{e}_3^s \end{pmatrix}}_{\mathbf{a}} + \lambda_3 \underbrace{\left(\begin{pmatrix} \mathrm{f}_1 \\ \mathrm{f}_2 \\ \mathrm{f}_3 \end{pmatrix} - \mathrm{f}_4 \begin{pmatrix} u \\ v \\ w \end{pmatrix} \right)}_{\mathbf{b}} = \mathbf{0}. \tag{7.15}$$

This is a 3D Parallel Vectors problem. The reformulation $\mathbf{a}||\mathbf{b}$ is equivalent to the coplanarity of the vector fields $\mathbf{p}, \mathbf{e}^s, \mathbf{f}$, and hence $\mathbf{a}||\mathbf{b}$ is satisfied exactly at the cores of swirling particle motion in unsteady flow fields. With this reformulation at hand we can use the powerful extraction techniques available for the Parallel Vectors operator.

The resulting structures are lines sweeping over time, i.e., surfaces in the 4D space-time domain. Bauer et al. [BP02] use a marching-cubes-like algorithm to extract these surfaces. In the following we present a method for tracking these cores using the concept of Feature Flow Fields. In fact, this works for two arbitrary time-dependent 3D vector fields $\mathbf{w}_1$ and $\mathbf{w}_2$. Based on the ideas of section 3.1.2 and following the notation used there, we want to extract PV surfaces in $\widetilde{D}$, the 4D space-time domain. To do so, we need two 4D feature flow fields $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$: One for following a PV line in space, the other one for following it in time. The first one can easily be identified as

$$\widetilde{\mathbf{f}}(x,y,z,t) = \begin{pmatrix} \mathbf{f}(x,y,z,t) \\ 0 \end{pmatrix} \tag{7.16}$$

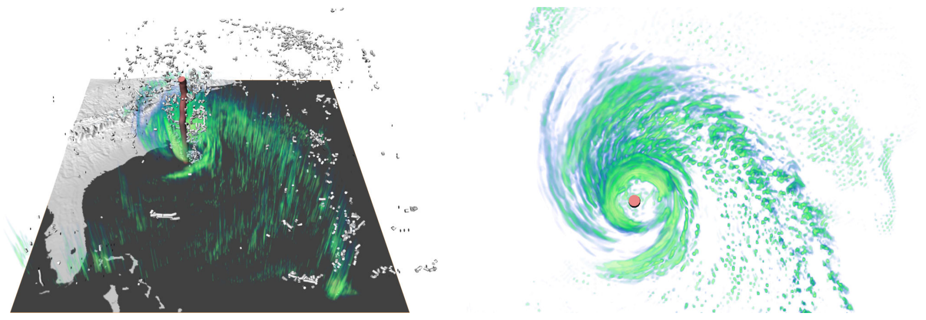
where $\mathbf{f}$ is defined in (3.2). It gives a PV line at a certain time level, i.e., all points on a stream line of $\widetilde{\mathbf{f}}$ have the same t-value. The evolution in time of a PV line should be covered by the 4D feature flow field $\widetilde{\mathbf{g}}$. Keeping in mind that PV structures in $\widetilde{D}$ are surfaces, a family of different $\widetilde{\mathbf{g}}$ could be chosen such that each linear combination of $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$ is a FFF. Among them, we choose the $\widetilde{\mathbf{g}}$ with $\widetilde{\mathbf{f}} \perp \widetilde{\mathbf{g}}$. This gives a unique $\widetilde{\mathbf{g}}$ (except for scaling). We obtain

$$\widetilde{\mathbf{g}}(x,y,z,t) = \begin{pmatrix} \mathbf{h} \times \mathbf{f} \\ ||\mathbf{f}||^2 \end{pmatrix} \quad \text{with} \quad \mathbf{h}(x,y,z,t) = \begin{pmatrix} \det(\mathbf{s}_x, \mathbf{s}_t, \mathbf{a}) \\ \det(\mathbf{s}_y, \mathbf{s}_t, \mathbf{a}) \\ \det(\mathbf{s}_z, \mathbf{s}_t, \mathbf{a}) \end{pmatrix}. \tag{7.17}$$

Using $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$ we can extract and track all locations fulfilling $\mathbf{w}_1||\mathbf{w}_2$ based on zero detection, stream line and surface integrations. The concrete algorithm follows the ideas presented in section 5.3, where we also used two FFF for extracting the skeleton of two-parameter-dependent vector fields. Details including a discussion of possible bifurcations of PV lines can be found in [TSW$^+$05].

Applications

In the following we extract cores of swirling motion from a number of flow fields. The computation times are reasonable low: in our implementation, a single time step of



(a) Complete extraction result prior to filtering. Lines shorter than a threshold are depicted in gray.

(b) Orthographic view from top showing the dominant particle core (red) in the center of the clouds.

Figure 7.12: Hurricane Isabel data set at $t = 33.5$. Shown are the dominating swirling particle core line (red) and a volume rendering of the cloud moisture mixing ratio.

dimension 128^3 is processed in a single thread in about 15 seconds on an AMD64 X2 4400+.

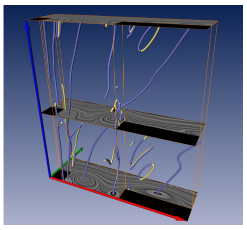
In 3D unsteady flows, cores of swirling motion are lines sweeping over time, i.e., 4D surfaces. One may use semi-transparent surfaces to encode past and future. However, for complex data sets these surfaces might contain self intersections. We found that displaying the core lines only – at a certain time step or in an animation – results in clearer visualizations in most cases.

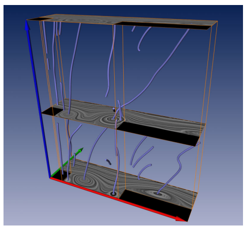
As Peikert et al. [PR99] already pointed out, one expects core lines to point in direction of the flow field, but the parallelity condition $\mathbf{a}||\mathbf{b}$ does not ensure this. Indeed, the solution lines can be orthogonal to the input vectors. Whenever this is not desired, one can filter the output such that only lines are displayed that do not exceed a defined threshold angle towards **a**, or **b**. In our implementation we use the angle between the solution lines and **e** as a criterion.

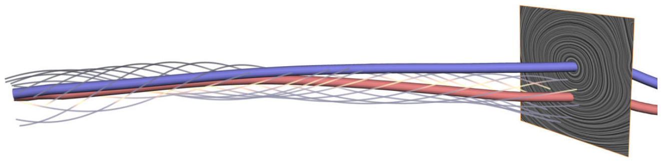
Note that the extraction of core lines of swirling motion is nonlinear in general, since eigenvectors do not depend linearly on the input fields. While this makes the extraction using linear techniques more difficult in general, we found that filtering the resulting lines by length was sufficient to rule out the nonlinearity.

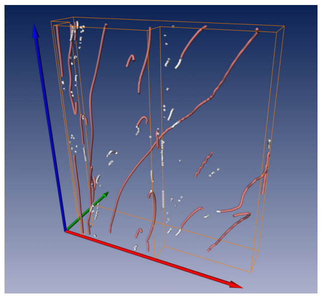
In Figure 7.12 we extracted the swirling particle cores of the Hurricane Isabel data set from the IEEE Visualization 2004 contest. This is a complex 3D time-dependent data set produced by the Weather Research and Forecast (WRF) model, courtesy of the U.S. National Center for Atmospheric Research (NCAR) and the U.S. National Science Foundation (NSF). Figure 7.12a shows the unfiltered extraction result at $t = 33.5$ consisting of 1533 core lines. We have chosen to filter all lines shorter than 10% of the diagonal of the bounding box. The result is the single swirling particle core line in the eye of the hurricane – verified by the volume rendering of the cloud moisture (Figure 7.12b).

Figure 7.13 shows an unsteady flow over a 2D cavity. This data set was kindly provided by Mo Samimy and Edgar Caraballo (both Ohio State University) [CSJ] as well as Bernd R. Noack and Ivanka Pelivan (both TU Berlin). 1000 time steps have been simulated using the *compressible* Navier-Stokes equations. The data is almost periodic, with a period of about 100 time steps in length, and only the first 100 time steps are shown.


(a) Time evolution of critical points.


(b) Swirling stream line cores.


(c) Close-up of a swirling particle core (red) in comparison to a swirling stream line core (blue).


(d) Swirling particle cores as identified by our new approach. Short, filtered lines are depicted in gray.

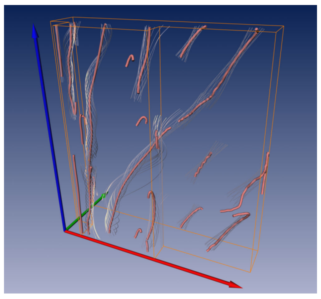
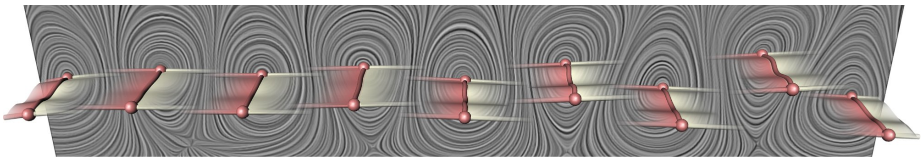

(e) Illuminated path lines verify that our cores are centers of swirling particle motion.

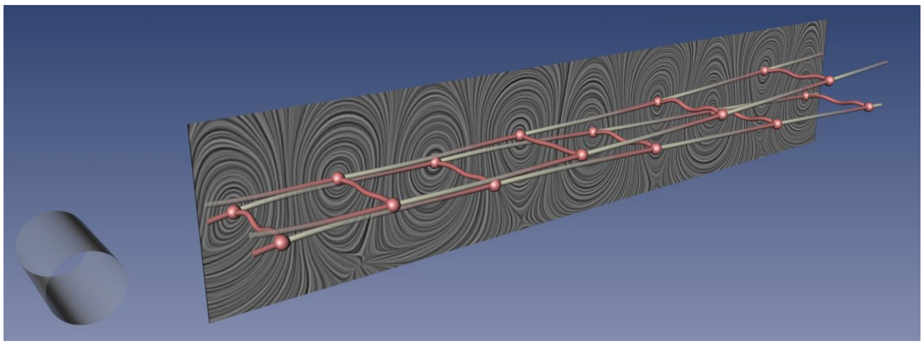
Figure 7.13: Unsteady flow over a 2D cavity. Red and green axes span the spatial domain, the blue axis denotes time.

As mentioned above, the Parallel Vectors operator can be used to track certain critical points (foci and centers) over time. Figures 7.13a-b exemplify this: the blue lines denote swirling motion of stream lines – once extracted by tracking critical points using Feature Flow Fields and once by applying the Parallel Vectors operator. Both results coincide very well. Note that additionally figure 7.13a shows tracked saddle points as yellow curves. Figure 7.13c stresses again the difference between swirling particle and stream line cores: the blue swirling stream line core goes through the center of spiraling stream lines at a specific time step (shown as LIC plane), but it does not lie in the center of spiraling path lines (shown as illuminated lines). Since unsteady motion is described by path lines, existing approaches fail to capture swirling motion in unsteady flows correctly: they are based on stream lines. Our approach captures this behavior correctly as shown by the red swirling particle core. Figure 7.13d shows the 181 extracted particle core lines, where the majority (154) is shorter than 3.5% of the diagonal of the bounding box and has been filtered accordingly. Figure 7.13e shows the filtered result.

Figures 7.14 and 7.15 demonstrate the results of our methods applied to a flow behind a circular cylinder. The data set was derived by Bernd R. Noack (TU Berlin) from a direct numerical Navier Stokes simulation by Gerd Mutschke (FZ Rossendorf). It resolves the so called 'mode B' of the 3D cylinder wake at a Reynolds number of 300 and a spanwise wavelength of 1 diameter. The data is provided on a $265 \times 337 \times 65$ curvilinear grid as a low-dimensional Galerkin model [NE94]. The flow exhibits periodic vortex shedding leading to the well known von Kármán vortex street [ZFN$^+$95]. This phenomenon plays an important role in many industrial applications, like mixing in heat exchangers or mass flow measurements with vortex counters. However, this vortex shedding can lead to undesirable periodic forces on obstacles, like chimneys,



(a) Evolution of the core lines over time is tracked by our algorithm and depicted using transparent surfaces. Red color encodes the past while gray shows the future.



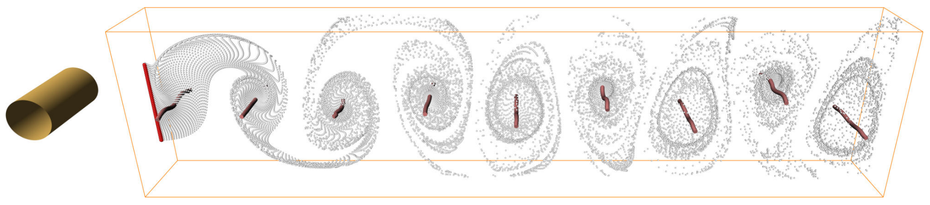
(b) The extracted seeding lines elucidate the alternating evolution of the vortical structures in transverse direction.

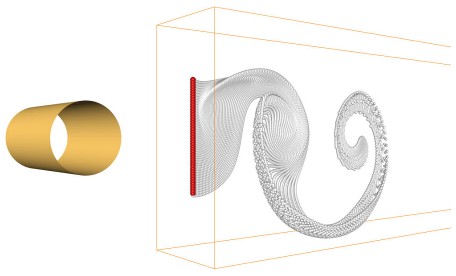
Figure 7.14: 3D unsteady flow behind a cylinder. Shown are swirling stream line cores in an appropriately chosen frame of reference.

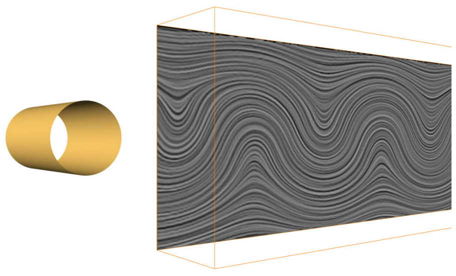
buildings, bridges and submarine towers.

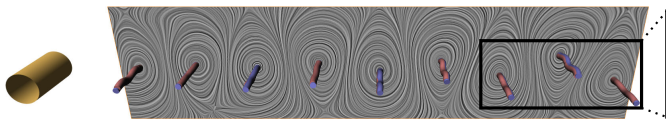
Figures 7.15a-b show particles seeded at a vertical line on the left-hand side of the bounding box. Due to the periodic vortex shedding these particles form patterns of swirling motion after some integration steps – a clear indication of the von Kármán vortex street. These patterns perfectly match up with the cores of swirling particle motion (red) extracted using our method (filtered by angle criterion with $|\cos(\ell, \mathbf{e})| < 0.3$ and by length with 0.1%). Figure 7.15c shows stream lines at a certain time step as depicted by the LIC plane: existing approaches based on stream lines have to fail to detect the von Kármán vortex street here, since the original frame of reference does not exhibit any spiraling stream lines. However, our method is based on the behavior of path lines and captures the vortex street correctly. Stream line based approaches are able to capture the features only if one chooses a reference frame matching their convection velocity. In this frame of reference, swirling motion of stream lines is present and its cores can be extracted by the method of Sujudi/Haimes. This has been done in figure 7.15d by applying a priori knowledge [ZFN$^+$95]. As a reference, the swirling particle cores (red) are also displayed. The extracted structures are very close to each other (Figure 7.15e). However, our method is able to extract these features in the original frame of reference without a priori knowledge.

Figure 7.14 shows swirling stream line cores in the same appropriately chosen reference frame as above. These visualizations exemplify that our extraction method is able to track PV lines over time. The time component of the resulting 4D surfaces has been encoded into transparency and color as shown in figure 7.14a: the surface fades out with increasing temporal distance to the current time step, and the color denotes the sign of that distance, i.e., red denotes the past and gray the future. This way the evolution of the core lines becomes visible in a single static image. Figure 7.14b shows


(a) Swirling particle cores (red lines) at a certain time step. The additionally shown particles verify that our core lines are at the centers of swirling particle motion.


(b) Particles are injected constantly at the vertical line on the left-hand side of the bounding box and advected over time. They form patterns of swirling motion indicating the von Kármán vortex street.


(c) Stream lines at a certain time step visualized using a LIC plane. Swirling motion of stream lines can not be observed in the original frame of reference.


(d) In an *appropriately chosen* frame of reference, swirling motion of stream lines is present (indicated by the LIC plane). The extracted swirling stream line cores (blue) are displayed together with the swirling particle cores (red) of the same time step.

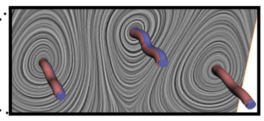
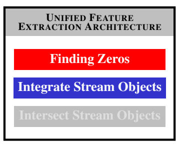

(e) Close-up of (d). Although extracted with different methods in different reference frames, the extracted lines are very similar.

Figure 7.15: 3D unsteady flow behind a cylinder. Existing stream line based approaches fail to capture swirling motion cores in the original frame of reference. Our new path line based method is able to extract such features without a priori knowledge.

the footprints on the boundary of all core lines for all time steps. This yields insight into the evolution of the vortical structures – elucidating the alternating nature of the vortex shedding in the von Kármán vortex street.

7.2.2 Galilean Invariant Vortex Core Lines



A variety of vortex definitions is not based on the velocity field nor the behavior of its characteristic curves, but on the properties of certain derived scalar quantities. These scalar fields indicate vortex activity in regions where they exhibit a certain value range. As an example, low or minimal pressure indicates a vortex – this is especially interesting in experimental setups where pressure can be actually measured. Hence, a vortex definition based on pressure allows comparisons between a simulation and a corresponding experiment.

vortex region quantity	vortex range	vortex core type
p	$[0,\infty)$	valley
$\|\omega\|$	$(0,\infty)$	ridge
Δ	$(0,\infty)$	ridge
Q	$(0,\infty)$	ridge
λ_2	$(-\infty,0)$	valley

Table 7.2: Vortex region quantities pressure p, vorticity ω, rotation strength Δ from [SP03], Q-criterion and λ_2 criterion with the value range in which they indicate vortices. Vortex cores according to definition 2 are either ridges or valleys as shown in the third column.

Examples of such vortex region quantities are listed in table 7.2. Most of these quantities have already been discussed in section 2.1.3 except for rotation strength Δ [SP03, CPC90]. It is linked to the intuitive understanding that a vortex exhibits spiraling stream lines with respect to some specific reference frame. Within this reference frame, the flow pattern is dominated by the Jacobian $\mathbf{J}$. If $\mathbf{J}$ has a conjugate pair of complex eigenvalues, the flow locally spirals in a plane corresponding to those eigenvectors. Δ is then defined as the magnitude of the imaginary part of those complex conjugate eigenvalues. So large values of Δ indicate strong spiraling patterns within the right reference frame. Where $\Delta = 0$, no such reference frame can be found. By considering the orientation of the corresponding eigenbasis, a rotation angle $\varphi \in (-\pi, \pi)$ can also be extracted. When $\varphi > 0$, the flow spirals counterclockwise around the eigenvector corresponding to the real eigenvalue, if $\varphi < 0$, clockwise.

These vortex region quantities have in common that they are Galilean invariant, i.e., they are invariant under adding constant vector fields. Furthermore, they have in common that vortex activity is locally maximal where the scalar field becomes locally extremal. Hence, the core line of a vortex can be defined as an extremum line of such a Galilean invariant vortex region quantity. We give the following:

Definition 2 *Let s be a Galilean invariant vortex region quantity. In regions where s identifies a vortex, a* **Galilean invariant vortex core line** *with respect to s is defined as a*

$$\left.\begin{matrix}\textit{ridge line}\\ \textit{valley line}\end{matrix}\right\} \textit{ of s if } \left\{\begin{matrix}\textit{large}\\ \textit{small}\end{matrix}\right.$$

values of s indicate a vortex.

Table 7.2 also shows whether so-defined vortex core lines are ridges or valleys of the respective quantity.

A vortex analysis based on definition 2 has a number of advantages:

- In contrast to an isosurface extraction or volume rendering of the respective quantity, the extraction of extremum lines is parameter free in the sense that their definition does not refer to a range of values. This eliminates the need of choosing certain thresholds or transfer functions. In other words, it objectifies the analysis and the extraction can easily be applied without user interaction, for instance as a batch job prior to visualization or during the simulation. See figure 7.16.

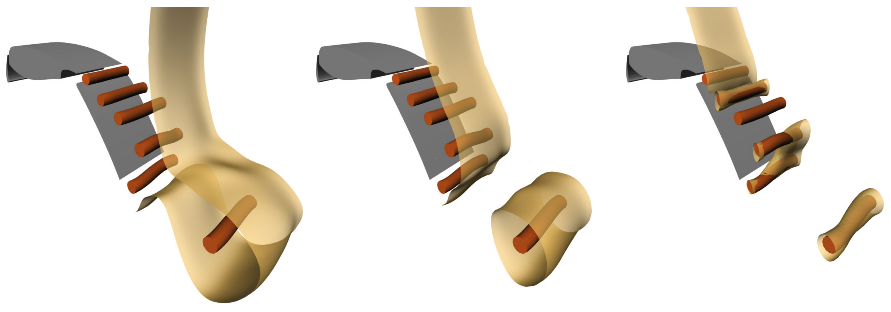


Figure 7.16: Minimal lines of pressure (red) and pressure isosurface (yellow) at the flow around an airfoil (data courtesy of B. Günther, TU Berlin). Shown is the same time step, but three different isovalues for the isosurface extraction. A precise vortex analysis using isosurfaces is difficult since they may break up along vortices or may contain several independent vortices. In contrast to this, extremum lines correctly identify location and extent of the vortices.

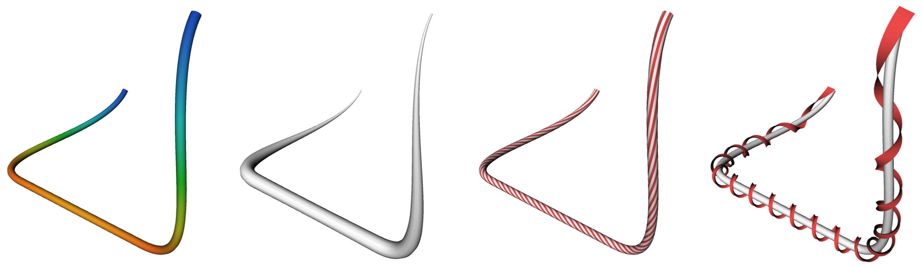
- Extremum lines correctly identify location and extent of the vortices in contrast to isosurfaces, which may break up along vortices or may contain several independent vortices. See figure 7.16.
- In contrast to a volume rendering, extremum lines allow for a quantitative analysis since they are well-defined geometric objects. Hence, several characteristics can be *measured*: the extent of a vortex, its distance to the body (e.g. an airfoil), its temporal evolution, its life time, and much more. This gives rise to a statistical analysis.
- In contrast to the cores of swirling motion as described in section 7.2.1, the core lines based on definition 2 are Galilean invariant.

Minimal lines of pressure have already been considered as vortex core lines prior to this thesis by Miura et al. [MK97]. This approach was applied locally only and resulted in disconnected line segments. In contrast to this, our method results in continuous lines. Furthermore, we consider arbitrary vortex region quantities.

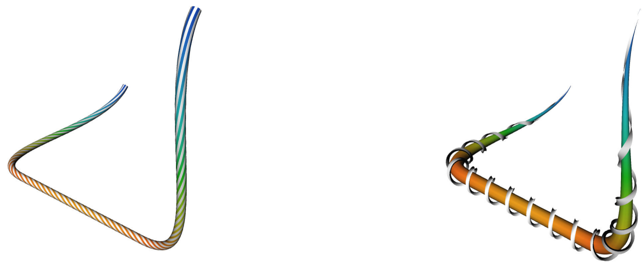
Several notions of extremum lines have been developed in the literature. In the next section 7.2.3 we will discuss how to extract such structures using topology. Here, we use the height ridge definition detailed in [Ebe96], which is a one dimensional generalization of the well known zero dimensional notion of an extremum point. We choose this definition as it requires just second derivatives of the vector field rather than fourth order derivatives like ridge definitions that are based on curvature extrema, see [EGM+94] for a thorough introduction and comparison of several ridge line extraction schemes and [KvD93] for a historical survey. An extremum line can be defined as a height ridge using the following definition cited from [Ebe96]:

Definition 3 *Let s be a scalar field, ∇s its gradient and Hs its Hessian with eigenvectors c_1, c_2, c_3 and corresponding eigenvalues $\gamma_1 \leq \gamma_2 \leq \gamma_3$.*

1. *Then a ridge line consists of all points $\mathbf{x}$ where*
   - $a := (\nabla s)c_1 = 0,$



(a) Color. (b) Scale. (c) Twist. (d) Orbit.



(e) Composed: Color & Twist. (f) Composed: Color & Scale & Orbit.

Figure 7.17: Different approaches to encoding a scalar value into the representation of a line.

- $b := (\nabla s)c_2 = 0$,
- $\gamma_2 < 0$.

2. *Valley lines of s are defined as ridge lines of* $-s$.

In order to extract height ridges, we use the concept of Feature Flow Fields by applying the algorithms for Finding Zeros and Integrating Stream Lines of the Unified Feature Extraction Architecture (see section 3.3). This involves two steps: first, certain points are extracted that lie on the extremum lines of interest. Those points are used as seed points in the second step, where the ridges are extracted as field lines of a derived vector field.

Assuming a point $\mathbf{x} \in \mathbf{R}^3$ fulfills $a = 0$ and $b = 0$, the tangent direction of the ridge through $\mathbf{x}$ is given as the vector product of the gradients of a and b, i.e., the desired FFF is $\mathbf{f} := \nabla a \times \nabla b$. The seed points for a FFF integration can be found as the locations where (a,b) becomes zero on two-dimensional subsets of the domain, e.g., the boundaries of the domain or the faces of the underlying grid structure. This reduces the problem to finding roots of a function $\mathbf{R}^2 \to \mathbf{R}^2$. For this setting, several Newton solvers can be applied, involving further differentiation. As $a = (\nabla s)c_1$ and $b = (\nabla s)c_2$ already involve second order derivatives of s, we favored a gradient-free minimization of the positive function $(a^2 + b^2)$ which turned out to be more stable. We used the method described in [Ebe96] based on Powell's search [Pow64] and inverse parabolic interpolation [PFTV91].

See [SWH05b] for more details on the extraction including a discussion of interpolation issues and robust derivative computation.

To visualize the resulting line structures, we use cylindrical meshes and encode different scalar values into their representation. Figure 7.17 illustrates this. In figures

7.17a-b we color or scale the cylinder according to the mapped values. Figures 7.17c-d encode sign and strength of a rotational behavior, either by using colored stripes on the cylinder itself or by placing a spiral shape around it. Our implementation allows us to combine these four variations as shown in figures 7.17e-f. Note that not all possible combinations produce expressive results. Especially the usage of an orbit (figure 7.17d) tends to yield cluttered visualizations in more involved settings.

While those kinds of representing a line are quite common, we are still left with finding appropriate measures to be mapped onto the vortex core lines. Jiang et al. [JMT02] depict spiraling stream lines around vortex core lines defined by swirling motion (see section 7.2.1). As we are concerned with Galilean invariant vortex core lines here, this approach is not directly applicable: stream or path lines do not necessarily swirl around these cores. Sato et al. [SP03] extract and display vortex hulls similar to isosurfaces around a vortex core line.

We propose the following measures to be used for an iconic representation of Galilean invariant vortex core lines:

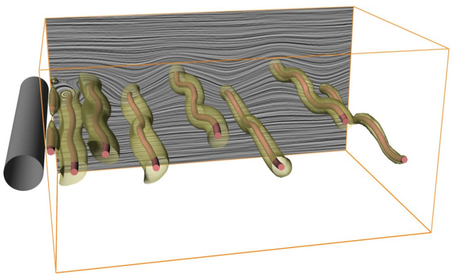
- *Strength/Value of vortex region quantity s:* Our vortex core lines are linked directly to a vortex region quantity s and their extremum property with respect to s ensures, that no regions indicating stronger vortex activity exist away from the extracted features. Furthermore, the value of s varies along a line. To distinguish between (parts of) core lines with different vortical activity, the value of s should be encoded in the line representation. We found coloring and scaling most suitable for this.

- *Sign of rotation angle φ:* As shown above, the rotation angle φ is derived from the Jacobian of the vector field. Its sign gives the direction of rotation of a vortex. As a visual encoding for this, the usage of color, twist or an orbit seems to be most appropriate.

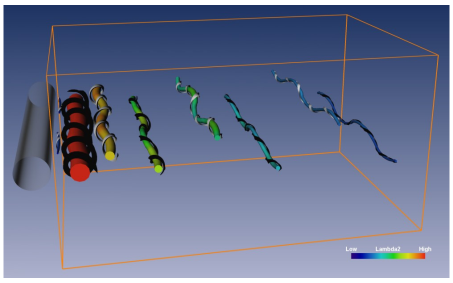
- *Strength of rotation Δ:* This measure indicates the strength of spiraling patterns in the right reference frame. We found the usage of color, twist or an orbit most suitable for this.

We apply these visualization strategies in different combinations in the following examples.

Figure 7.18 visualizes a snapshot of a transitional wake behind a circular cylinder [ZFN$^+$95]. This data set was derived from a direct numerical simulation of the Navier-Stokes equation by Bernd R. Noack (TU Berlin). It is given on a $88 \times 106 \times 20$ uniform grid. The data resolves the so-called 'mode A' of the 3D transition at a Reynolds number of 200 and at a spanwise wavelength of 4 diameters. We extract the vortices in this example using the λ_2 criterion. Figure 7.18a shows that methods based on swirling motion fail in this context: since only a single time step is available, we have to resort to the motion of stream lines (and not path lines as in section 7.2.1). As it can be seen in the LIC plane, the stream lines in this original frame of reference do not show spiraling patterns as already known from the von Kármán vortex street. However, the λ_2 criterion – depicted using isosurfaces and the corresponding valley lines – is able to capture the vortices in the original frame of reference and using a single time step only. The chain of vortices with their alternating orientation of rotation is clearly depicted in figure 7.18b due to the usage of spiraling orbits. This is a major property of the von Kármán vortex street. Furthermore, it can be seen that downstream the vortices loose their strength.

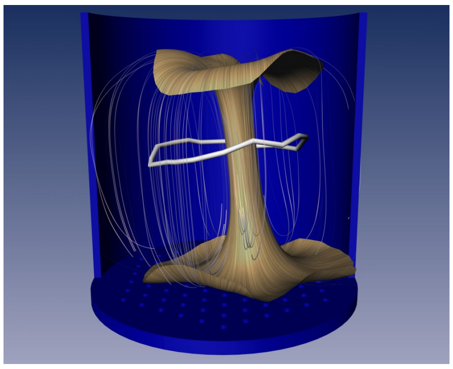


(a) Valley lines (red) and isosurfaces (yellow) of λ_2. The additionally shown LIC plane shows that the vortices cannot be observed by means of swirling motion in the original reference frame.

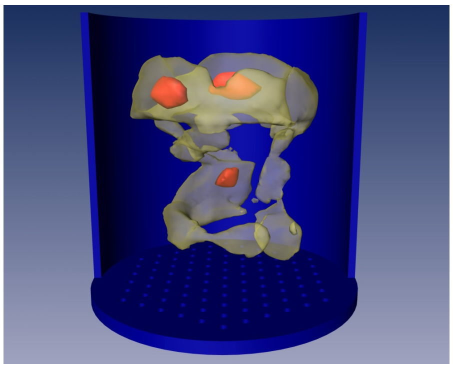


(b) Vortex core lines colored and scaled according to λ_2. Red / blue color indicates strong / weak vortex activity. φ is encoded into color and spiral direction of the orbits.

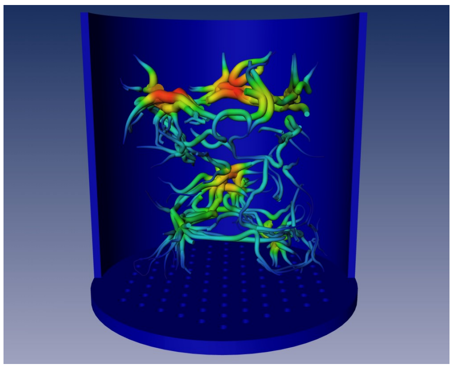
Figure 7.18: Flow behind a circular cylinder. Galilean invariant vortex core lines extracted as valley lines of λ_2.



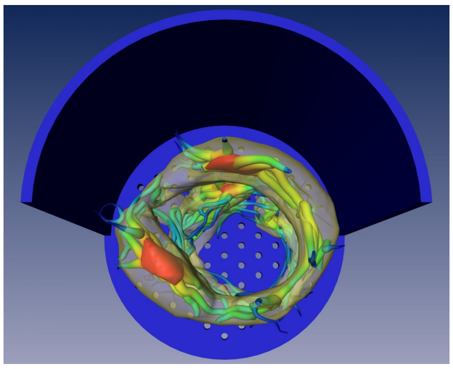
(a) Visualized using illuminated field lines [ZSH96] and a LIC-textured stream surface [BSH96]. Swirling stream line cores following [SH95b] displayed as gray lines.



(b) Isosurfaces of λ_2.



(c) Galilean invariant vortex core lines.



(d) Comparison between isosurfaces and core lines. View from top.

Figure 7.19: Bubble chamber. Vortex core lines extracted, colored and scaled according to λ_2. Same colormap as in figure 7.18b.

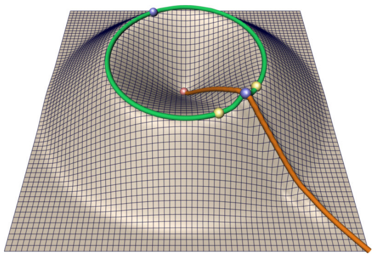
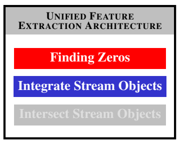


Figure 7.20: 2D terrain exemplifying different extrema definitions. The closed green line is a watershed separatrix. Both the green and the orange line conform to the height ridge definition.

Figure 7.19 shows the geometry of a bubble chamber and its interior flow. The flow has been measured experimentally on a $11 \times 11 \times 10$ uniform grid by a biplanar x-ray angiography in a biofluidmechanics laboratory. The bubble chamber is used as a biochemical reactor. Air injection into the liquid through holes in the floor plate is used to improve the reaction. The dataset was provided by Axel Seeger, Biofluidmechanics Lab, Charite Berlin. Figure 7.19a shows the swirling stream line core line according to [SH95b] around which the flow spirals. Figure 7.19b shows isosurfaces of λ_2 corresponding to different isovalues. In figure 7.19c, the vortex core lines with respect to λ_2 extracted by our method are shown, sized and colored corresponding to λ_2. Figure 7.19d is a combination of figures 7.19b and 7.19c looking into the bubble chamber from above. This figure clearly shows that our approach yields vortex core lines in the center of the considered vortex region quantity.

7.2.3 Vortex and Strain Skeletons



In the previous section we advocated the use of extremum lines of certain derived scalar fields as vortex core lines and extracted them using the height ridge definition. In this section we propose a different extraction scheme based on the topology of the scalar field: extremum lines can be found as the separation lines of the topological skeleton of the gradient. Furthermore, the other elements of the topological skeleton like critical points and separation surfaces also have a meaning in certain settings – leading to the notion of vortex skeletons. Since the extraction is based on the gradient of the considered scalar field (which is a vector field), the algorithms of the Unified Feature Extraction Architecture can be applied as already described earlier. While the extraction of height ridges needs second order derivatives, the separatrices can be extracted using first order derivatives only – leading to a more robust extraction scheme.

The topology of a 2D scalar field is closely linked to watershed lines in a 2D terrain (cf. figure 7.20): at certain line structures, rain water separates in the sense that nearby water assembles in different valleys. Those maximum lines partition the domain into valleys. Within valleys all water flows towards the same minimum. Similarly, the domain is partitioned into hills separated by minimal lines called watercourses. On hills all water runs down from one maximum.

The generalization to 3D is straightforward. Here the watersheds are surfaces, and additional one-dimensional separatrices come into play, which are lines of minimal/maximal scalar value. Under the assumption that the examined scalar field is an indicator of vortex activity (e.g. as listed in table 7.2) these lines denote vortex core lines following definition 2.

The topological skeleton of a scalar field is called Morse-Smale complex. A num-

scalar field s	gradient vector field $\mathbf{v} = \nabla s$
minimum	source
1-saddle	repelling saddle
2-saddle	attracting saddle
maximum	sink
maximum line	unstable 1D manifold, repelling separation line
minimum line	stable 1D manifold, attracting separation line
maximum surface	unstable 2D manifold, repelling separation surface
minimum surface	stable 2D manifold, attracting separation surface

Table 7.3: Terminology of the topological elements in 3D scalar and vector fields. Note that all critical points of a gradient vector field are of node-type, i.e., all eigenvalues are real and rotational stream line behavior cannot be observed. For example, foci and center cannot appear in such fields. In the 2D setting, there are no surface features and no distinction between different saddles.

ber of different extraction strategies can be applied:

- The *watershed transformation* comes from the field of image analysis and is based on the analogon of terrains, rain water and hills/valleys as described above. It is a common alternative to height ridges for the extraction of extremum lines [Ebe96, Soi99]. A thorough overview of different definitions and algorithms can be found in [RM00]. Most of these algorithms result in a discrete segmentation of the domain into label regions and the watersheds itself can be extracted as the borders between these regions [HSSZ97].

- Edelsbrunner et al. [EHNP03] extract and simplify the Morse-Smale complex in a discrete manner on triangulated surfaces based on so-called *simulated differentiability*. This approach has been extended to 3D scalar fields by Gyulassy et al. [GNP$^+$05, GNPH07].

- Based on the *discrete morse theory* of Forman [For98, For02], Cazals et al. developed an algorithm for the extraction of the Morse-Smale complex on triangulated surfaces [CCL03].

- If the scalar function is differentiable, its topology can also be obtained as the *vector field topology* of its gradient.

The first three schemes allow for a derivative-free extraction of the topological skeleton, but the result is a discrete subset of the input grid, i.e., lines and surfaces are not smooth. The results of the fourth scheme are smooth. We choose this approach for the remainder of this section since it is an application of the concepts and algorithms developed in this thesis, and refer the reader to [SWTH07] for a discussion of the application of the discrete watershed transformation in this context.

The elements of the topological skeletons of a scalar field s and its gradient vector field $\mathbf{v} = \nabla s$ correspond to each other as shown in table 7.3. It is easy to see that the separation lines of $\mathbf{v}$ are the desired extremum lines of s. See [SWTH07] for a discussion on how these lines can be compared with height ridges.

This extraction scheme can be applied to all kinds of scalar fields in order to extract extremal features. However, some scalar fields may have the property that both minimal and maximal features have a physical interpretation. This is the case for flow fields, where the decomposition of the flow field gradient $\mathbf{J}$ into the strain tensor $\mathbf{S}$ (2.19) and the vorticity tensor Ω (2.20) obeys the following duality: vortical activity is high in regions where Ω dominates $\mathbf{S}$, whereas strain is characterized by $\mathbf{S}$ dominating Ω. Two quantities utilize the decomposition of $\mathbf{J}$ to identify not only vortices but also strain regions. The Okubo-Weiss criterion Q (2.21) identifies regions with

- $Q > 0$ as vortex regions and
- $Q < 0$ as strain regions.

In particular, maximum lines of Q are vortex core lines, and minimum lines denote the cores of high strain. The topological skeleton of Q allows a concurrent analysis of both features in a consistent setting.

Haller has recently proposed the $\mathbf{M_Z}$-criterion that also discriminates vortex and strain regions in incompressible flows similar to the Q-criterion, but in contrast to this based on a Lagrangian analysis [Hal05]. The $\mathbf{M_Z}$-criterion is based on a strain analysis along path lines. Loosely spoken, Haller proves that path lines along which a certain strain acceleration tensor M is positive definite are of saddle type – so called *hyperbolic* lines of maximal strain. Based on this, vortices are defined as the opposite, i.e., path lines along which M is indefinite. Such structures are called *elliptic*. Since it is unlikely to find a path line that is completely hyperbolic or completely elliptic, the following qualitative property is used:

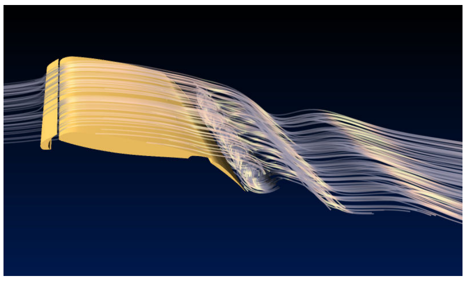
- The lower $\mathbf{M_Z}$, the more strain is present.
- The higher $\mathbf{M_Z}$, the more vortical behavior is present.

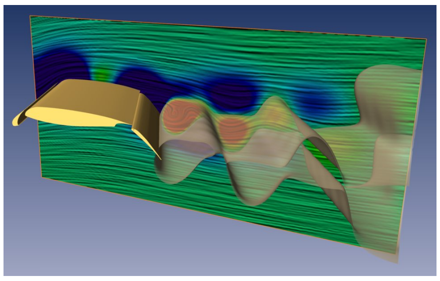
This is a duality property similar to the Okubo-Weiss criterion, and, again, the extremum lines of $\mathbf{M_Z}$ are the core lines of the respective regions. Note that the duality of vortex and strain activity is intrinsic to the $\mathbf{M_Z}$-criterion, as $\mathbf{M_Z}$ is solely based on a strain analysis. Hence, Haller defines a vortex as *lack of strain*. A more detailed introduction to $\mathbf{M_Z}$ and a discussion of its quite involved computation can be found in [SWTH07].

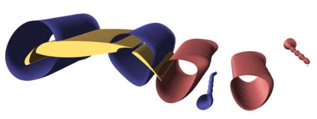
To the best of our knowledge, the $\mathbf{M_Z}$-criterion has not been used prior to this thesis in the Visualization and Computer Graphics community. Furthermore, a concurrent feature-based analysis of strain and vortex activity has not been done before.

Figure 7.21 shows the flow around a Swept-Constant-Chord-Half-model (SCCH) of an airfoil that was simulated by Bert Günther (TU Berlin) at a Reynolds number of 10^6 on a curvilinear block structured grid with 1.3 million cells. Due to the constant chord and periodic boundary conditions this is a 2.5D configuration. The sweep angle of the airfoil to the flow direction is 30° and the angle of attack is 6°. The turbulence was simulated by a combined URANS and DES approach. Figure 7.21d shows the line type structures of the vortex and strain skeletons of Q. Note that by our method, the collection of all extremal strain and vortex lines provide a good overview over the dataset, while the isosurfaces in Figure 7.21c miss the smaller features downstream.

In Figure 7.22 we applied our methods to a 3D time-dependent turbulent mixing layer. The velocity field has been computed with a pseudo-spectral direct numerical simulation by Pierre Comte, employing the computational domain and boundary conditions of [CSB98]. The Reynolds number is 100 based on the initial shear-layer thickness and convection velocity. The velocity ratio between the upper and lower stream is $3:1$ (Figure 7.22a). The data consists of 500 time steps of a $480 \times 48 \times 96$ uniform grid. Figure 7.22b shows the minimum lines of $\mathbf{M_Z}$ which correspond to lines


(a) Stream lines of the time-averaged flow around the airfoil.


(b) Isosurface $Q = 0$ and a LIC plane colored by Q.


(c) Isosurfaces of $Q < 0$ denoting strain (blue) and $Q > 0$ denoting vortex activity (red).

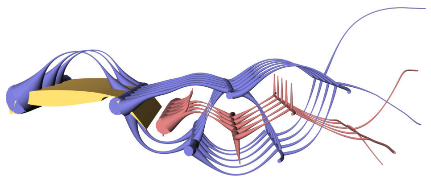
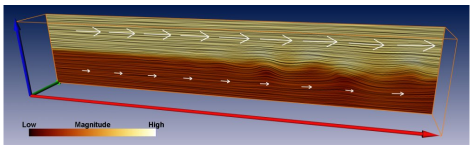
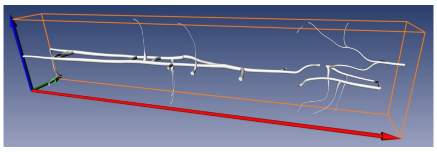
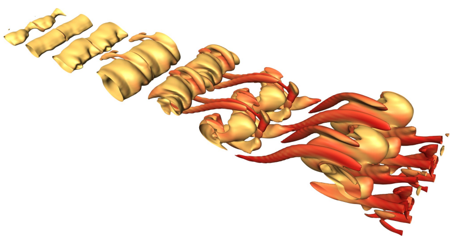

(d) Lines of maximal strain (blue) and maximal vortex activity (red) scaled by $|Q|$.

Figure 7.21: Flow around an airfoil. The lines of the vortex and strain skeletons give a complete overview of the location and extent of vortex and strain features in the flow, whereas the isosurfaces miss the smaller features downstream and give only a rough location for the larger features upstream.



(a) Flow visualized using LIC.


(b) Lines of maximal strain of $\mathbf{M_Z}$.


(c) Isosurface of Q.

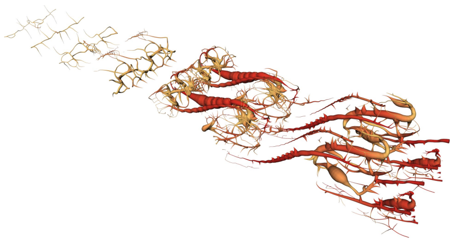

(d) Lines of maximal vortex activity scaled by Q.

Figure 7.22: Turbulent mixing layer. The lines of maximal strain as indicated by $\mathbf{M_Z}$ perfectly match up with the shear layer. The vortex skeleton of the Q-criterion elucidates the spatial evolution of Kelvin-Helmholtz vortices, vortex pairing, and the spanwise formation of streamwise rib vortices.

of maximal strain. It can clearly be seen that those structures lie in the shear layer which corresponds to intuition. In 7.22c isosurfaces of Q display the spatial evolution of Kelvin-Helmholtz vortices (primary vortex structures), vortex pairing, and the spanwise formation of streamwise rib vortices (secondary vortex structures). In 7.22d the vortex skeleton of Q is shown with lines scaled by the value of Q. The whole vortex structure of the flow can be seen at one view. In particular, our method is capable of resolving secondary vortex structures as well as the less vortical structures further upstream that are hidden by the isosurfaces in 7.22c.

7.3 Steering of other Visualization Techniques

Most visualization methods need to be parametrized in some way. For example, a prerequisite for the visualization of stream lines is a set of seeding points, or an isosurface visualization depends on the definition of an isovalue. Manual adjustment of such parameters can be a tricky and very subjective task for the user. An automated, or at least guided, feature-based definition of visualization parameters helps the user to analyze the data more objectively and target-oriented. In the following we give some examples where topology has been used to steer or guide other visualization techniques.

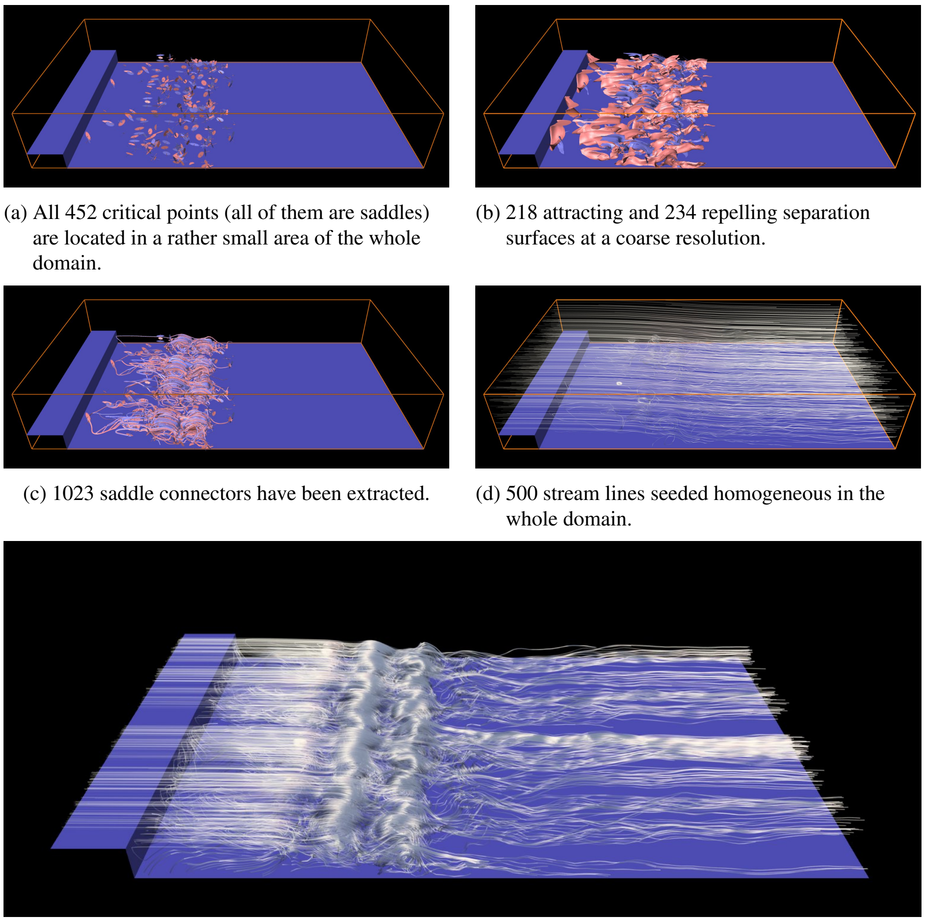
Figure 7.23 shows the transitional flow around a backward-facing step where we extracted critical points to define a set of seeding points for a stream line visualization. The flow field is obtained from a numerical simulation of Kaltenbach and Janke at a Reynolds number of Re_H=3000 based on oncoming velocity and on step height. The corresponding boundary conditions are described in [KJ00].

The data set contains 452 critical points which are visualized in figure 7.23a. Since the vector field is divergence-free, all of them are saddles. Figures 7.23b-c show the separation surfaces and the saddle connectors in addition to the critical points. As we can see there, the flow contains topological information only in a rather small part of the domain, but the topology is so complex that a direct depiction of these structures does not give an expressive and insightful visualization.

A straightforward stream line visualization is neither helpful, as shown in figure 7.23d. Here, stream lines have been seeded homogeneously over the whole domain. It turns out that the stream lines in the laminar parts of the flow hide the more interesting regions.

A key to an expressive visualization of this data set is the combination of both techniques: in figure 7.23e we made use of the extracted critical points and seeded stream lines close to them. One may place the seeds around a critical point using certain patterns considering its type as suggested by Verma et al. for 2D [VKP00] and Ye et al. for 3D vector fields [YKP05]. In figure 7.23e we seeded randomly around each critical point. The usage of topological information to steer the seeding of stream lines yields a far more expressive and insightful visualization than figure 7.23d, where stream lines have been seeded homogeneously over the whole domain. Note that both figures show the same number of stream lines. Figure 7.23e illuminates the coherent structures of this type of flow: the flow separates at the corner of the step. The resulting shear layer rolls up in two Kelvin-Helmholtz vortices. In the downstream direction, the streamlines form bundles due to secondary streamwise vorticity. The fluid experiences a small backward flow in the upstream region below the shear layer.

Figure 7.24 shows the flow behind a circular cylinder. Again, homogeneously distributed stream lines do not give much insight (figure 7.24a). The topology of this flow has been visualized in figures 4.11 and 4.12 on page 70. Since the topology is not too



(a) All 452 critical points (all of them are saddles) are located in a rather small area of the whole domain.

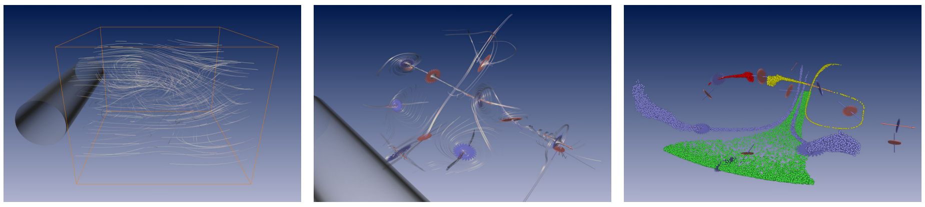
(b) 218 attracting and 234 repelling separation surfaces at a coarse resolution.

(c) 1023 saddle connectors have been extracted.

(d) 500 stream lines seeded homogeneous in the whole domain.

(e) 500 stream lines seeded near critical points.

Figure 7.23: Flow around a backward-facing step.



(a) Homogeneously distributed stream lines.

(b) Stream lines seeded near critical points.

(c) Particles seeded near critical points after some integration time.

Figure 7.24: Flow behind a circular cylinder. Stream lines and particles seeded close to the critical points. Data courtesy of Gerd Mutschke (FZ Rossendorf).

complex, it already gives a good insight into the flow behavior behind the cylinder. This can be increased in combination with other techniques as shown in figures 7.24b and 7.24c, where we have seeded stream lines and particles close to the critical points.

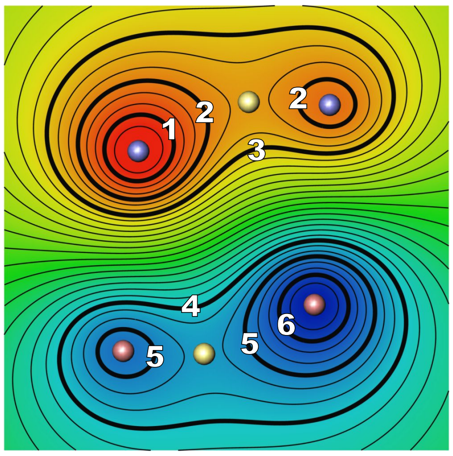
Figure 7.25 illustrates an application of topological methods to steer the visualization of isolines/isosurfaces (level sets) of scalar fields. Figure 7.25a shows a 2D scalar field with color coding and isolines, and its topological skeleton (Morse-Smale complex) is shown in figure 7.25b together with its gradient (see also section 7.2.3 for a discussion of the topology of scalar fields and its gradient vector fields).

The topology of a scalar field is closely linked to the properties of level sets: with decreasing isovalue, connected components (contours) of the level set are created at maxima, they join or split at saddles, and they disappear at minima (figure 7.25d). This can be used to build a so-called contour tree [BR63, TIS$^+$95, PCM03], which charts the relationship of connected components along the scalar value (figure 7.25c). The nodes of this tree correspond to the critical points: minima and maxima are leafs, saddles are branching points. The edges resemble the connectivity between the critical points. Among other things, a contour tree allows to compute the number of connected components and to distinguish between different components for each isovalue prior to the extraction of the level set itself. Carr et al. [CS03] propose a user interface where a contour tree is used to interactively manipulate an isosurface visualization. Similar approaches exist for volume rendering [TTFN05, TFT05].

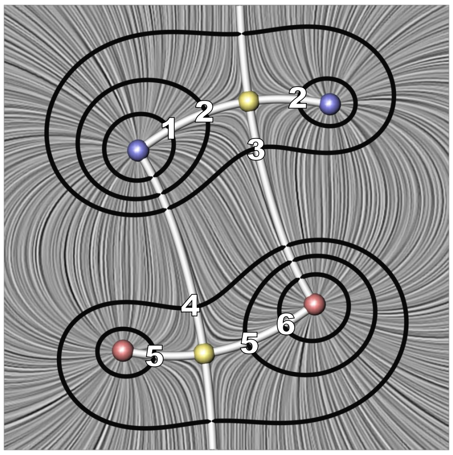
Another interesting link between levels sets and topology: every level set intersects a separatrix of the topological skeleton (cf. figure 7.25b). This is due to the fact that the set of all separatrices covers the complete value range of the scalar field – globally and locally, since each minimum/maximum is connected to at least one separatrix. A point on a separatrix with the desired isovalue can be used as a seed point for the extraction of the whole contour, since one point suffices to uniquely identify a contour. Hence, this property can be applied in combination with contour trees to accelerate the extraction of level sets [vKvOB$^+$97, CS03].

Figure 7.26 shows a snapshot of a 3D mixing layer where we used topological methods to augment an isosurface and volume rendering visualization. The velocity field has been computed with a pseudo-spectral direct numerical simulation by Pierre Comte, employing the computational domain and boundary conditions of [CSB98]. The Reynolds number is 100 based on the initial shear-layer thickness and convection velocity. The velocity ratio between the upper and lower stream is $3:1$. The velocity difference between both streams is responsible for the occurrence of vortices in this flow. To study them, we visualized the primary and secondary vortex structures using isosurfaces and volume rendering of the Okubo-Weiss parameter Q. In order to distinguish between primary and secondary structures, we pronounced the primary Kelvin-Helmholtz vortices using $Q\cos^2\theta$, where θ represents the local angle between the vorticity vector and the spanwise axis. This quantity is represented using a bright volume rendering. Similarly, the secondary rib vortices are accentuated with $Q\sin^2\theta$, shown as red isosurface. Using this local angle criterion allows to distinguish between primary and secondary vortex structures, and it ensures that the volume rendering dominates the isosurfaces in the respective areas and vice versa.

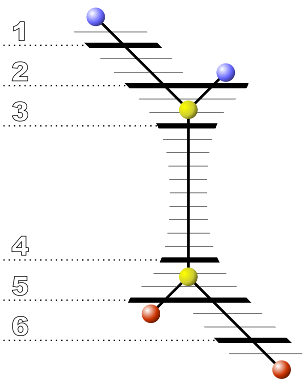
The primary vortices are separated from each other by regions where a saddle-like flow behavior can be observed in a frame of reference which moves with the convection velocity. In order to capture this flow behavior, we utilize topological methods known from 2D time-dependent vector fields – despite the fact that this is a single time step of a 3D flow. In fact, we search for all locations where the streamwise and transverse velocity components vanish, i.e., for critical points in a plane aligned with the



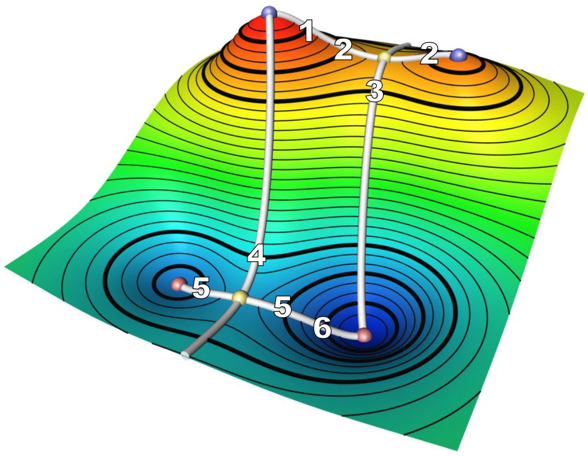
(a) Scalar field with isolines and minima (red), maxima (blue), and saddles (yellow).



(b) Topological skeleton of the gradient with sources (red), sinks (blue), and saddles (yellow).



(c) Contour tree with indicated isovalues.



(d) Terrain visualization including topological skeleton and isolines.

Figure 7.25: Properties of isolines can be inferred from the topological skeleton. An isoline may consist of different connected components, which are created at maxima, join or split at saddles, and disappear at minima (with decreasing isovalue). This behavior is recorded in a contour tree, where minima/maxima are leafs and saddles are branching points. Note that six different isovalues/isolines have been highlighted and labeled accordingly. Two of them (the 2nd and the 5th) consist of two connected components, which can also be observed in the contour tree where their corresponding isovalues intersect with two branches of the tree.

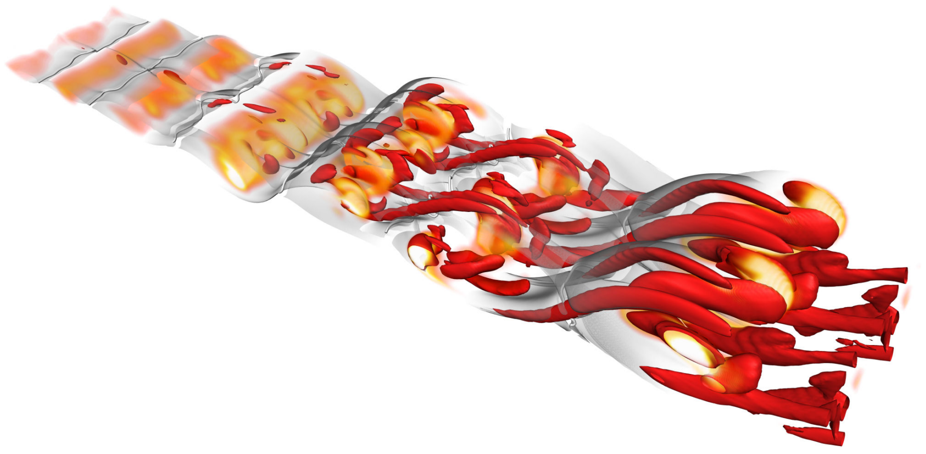


Figure 7.26: Coherent structures of a turbulent mixing layer elucidated using a combination of topology, isosurfaces and volume rendering.

streamwise and transverse axes. To do so, we consider the spanwise direction as the time/parameter axis, neglect the spanwise component of the flow and track the critical points as described for time-dependent vector fields in section 5.1.1. From the resulting set of lines we choose the tracked saddle points (the others are tracked centers denoting the vortices). They are shown in figure 7.26 as white curves. The stable and unstable manifolds of these quasi-saddles are shown by transparent surfaces generated by a stream surface integration. It turns out that these surfaces enclose the primary Kelvin-Helmholtz vortices. Tricoche et al. [TGK^{+}04] allow a similar analysis using the concept of so-called moving cutting planes.

The combined visualization of the mixing layer using topology, isosurfaces, and volume rendering elucidates the spatial evolution of Kelvin-Helmholtz vortices, vortex pairing, and the formation of spanwise rib vortices near the saddle points.

Chapter 8

Discussion

In this chapter we discuss the applicability of topological methods under certain aspects like their sensitivity to noise or application to turbulent flows. Furthermore, we study the topological complexity of 2D and 3D vector fields and prove a fundamental difference between both cases, which seems to be a reason for the fact that topological methods are still less common for 3D vector fields than for 2D. We start with considering the topology of other characteristic curves.

8.1 Topology of other Characteristic Curves

The main motivation behind topological methods is to segment a vector field into areas of similar flow behavior which is determined by observing the behavior of certain characteristic curves. While this thesis is mainly concerned with the behavior of stream lines, the topology of other characteristic curves will briefly be discussed here.

For steady vector fields there is only one important class of curves, stream lines, and the corresponding topological structures have been discussed in chapter 4.

For time-dependent vector fields, however, there are four classes of curves: stream, path, streak, and time lines. As already discussed in section 2.1.2, streak and time lines fail to have an important property: they are not locally unique, i.e., for a particular location and time there is more than one streak and time line passing through. This disqualifies them for topological considerations since their flow behavior cannot uniquely be assigned to a location.

Two important classes of curves remain for time-dependent vector fields: stream and path lines. Hence, two different kinds of topologies can be considered: a *stream line oriented topology* where areas are segmented which show a similar behavior of stream lines, and a *path line oriented topology* which does so for path lines.

Stream line oriented topology has been discussed extensively in the previous chapters and almost all topological methods developed in the visualization community fall into this category.

Contrary, only a few approaches have been published on the topology of path lines. A reason for this might be that the vector field $\mathbf{p}$ (2.13) (which describes the path lines of a time-dependent vector field $\mathbf{v}$ as tangent curves) does not have any zeros at all. Hence, a straightforward application of classical methods fails.

A second reason could be the impossibility of an infinite integration in $\mathbf{p}$. In fact, stream line based topology is based on the idea of infinite stream line integration: assuming no-slip boundaries, a forward integration ends in a sink, a backward integration in a source, or it is a closed stream line. The asymptotic behavior of stream lines is the foundation of their topology and it allows to group stream lines based on their connectedness to sources, sinks, or periodic orbits.

An infinite integration of path lines is usually not possible since the time dimension is typically bounded by t_{min} and t_{max}. Furthermore, since zeros are missing in $\mathbf{p}$, path lines will not converge to critical points. Hence, a statement about the asymptotic behavior of path lines in vector fields defined over a fixed time interval is not possible.

Two solutions to this can be given: considering time-periodic vector fields, or examining the path line behavior for a finite time integration.

For periodic vector fields, the time dimension is not restricted to a certain interval but can be extended to any time by periodically repeating the given field. Hence, an analysis of the asymptotic behavior of path lines becomes possible. In fact, many numerical flow simulations are actually periodic (or pseudo-periodic) flows. A number of such flows has been treated in this thesis, like the flow around a cylinder (page 87), the cavity data set (page 88), or the flow around an airfoil (page 145). As found by Shi et al. [STW$^+$06], periodic flows may contain *critical path lines* which repeat the same spatial cycle in every time period. These critical path lines are the basis of the topological segmentation: using certain Poincaré maps, all path lines are classified whether they converge to a critical path line in forward or backward integration respectively. Similar to critical points of a vector field, critical path lines can act as sources, sinks, or saddles.

Haller [Hal00, Hal01, Hal05] examines the finite time behavior of path lines in incompressible flows and distinguishes in the resulting scalar fields between different Lagrangian structures which are locally extremal. Approaches to visualizing the structures of [Hal00, Hal01] are given by Sadlo et al. [SP07] and Garth et al. [GGTH07]. However, a topological segmentation of the domain into regions of different path line behavior based on the vortex/strain duality can only be achieved using [Hal05] and the corresponding feature extraction method laid out in section 7.2.3 and published in [SWTH07].

8.2 Noise and Derivatives

Virtually all real world data sets contain a certain level of noise. While numerical simulations are usually rather smooth, a high noise level can be found in experimentally measured data sets.

All visualization techniques have to cope with this and it is not necessarily a problem as long as the visualization truthfully represents the content of the data set. In a sense, visualization can also serve as an evaluation tool of experimental measurement equipment or numerical simulation methods. However, it is desirable to visualize the underlying physics and not the obfuscating noise.

Some visualization methods are more sensitive to noise than others. Consider the integration of a stream line as it is for example shown in figure 3.4a on page 52: a small perturbation of the vector field suffices to alter the complete path of the stream line. All stream line based visualization techniques suffer from this. In fact, a LIC texture is as exposed to the noise as a topological skeleton, since both represent the very same: stream lines of a vector field.

However, as one can usually notice, a LIC texture is not too much affected by the change of all stream lines due to a perturbation. This can be explained using topology:

in most cases, a small perturbation does not alter the connectedness of a stream line to its source and sink, since this is a stable connector as already found in section 3.2 (figure 3.4b). In other words, the structure of the vector field remains the same (or at least it is similar) and therefore a LIC texture leaves a similar visual impression – although the geometry of all stream lines has been changed. This structural stability is represented by the topological skeleton: while a small perturbation slightly alters the locations of the critical points and the geometry of the separatrices, the graph structure between the critical points remains the same. This is only true as long as no new critical points appear due to the perturbation. In the case of new saddles, new separatrices will emerge. This is a sudden change and can perfectly be perceived by the human visual system, which is more sensitive to an appearance of new distinct geometric objects than to a perturbation of a texture. Therefore, a LIC visualization is usually regarded as being more robust against noise than a topological visualization, although new critical points and separatrices can be observed in the LIC texture as well.

In most real world data sets, noise becomes stronger in higher derivatives. This is a problem for most feature-based methods since they often depend on derivatives. There are several ways to deal with noise:

- **filtering of data sets**
  It is possible to smooth the data set by filtering it using e.g. a Gauss kernel. Experimentally measured flows are often time-averaged, if the underlying flow phenomenon is steady or at least almost steady. In this case, averaging is an arguable approach to get a smooth data set. In unsteady settings, such an averaging could be applied to a number of adjacent time steps using e.g. a Gauss kernel. Note that filtering the data set does not give control over the resulting topological or vortical structures.

- **filtering of extraction results**
  Applying feature extraction techniques to noisy data sets usually results in a high number of extracted features leading to cluttered, unintuitive visualizations. If it is possible to assign a certain importance or strength to each distinct feature (quantification), one could remove features below a certain user-defined threshold from the depiction. Another approach is to scale the depiction of the features according to their strength. In any case, filtering the extraction results is preferable to filtering the original data, since the remaining structures are actual features of the original data set and not of a smoothed version thereof.

- **higher order interpolation schemes**
  Features depending on higher order derivatives (like e.g. ridge lines discussed in section 7.2.2) may not be well resolved by linear or trilinear interpolation schemes. Higher order schemes have to be applied. Among those, approximation by quadratic super splines [RZNS04] provides a good trade-off between smoothness and speed.

8.3 Turbulent Flows

All visualization methods have a common weakness: the visualized data set needs to have structures which can be recognized and form some patterns. Structures and patterns is what users actually look for when visualizing data, and especially when extracting features. This observation leads to two worst case scenarios:

- There are (almost) no features in the data set.
- There is a high amount of features in the data set, but they are equally distributed and equally strong.

The first problem is discussed in more detail in the next section. Here, we want to comment on the second problem, where a high number of features does not form recognizable patterns.[1]

Turbulent flows are an example for this. They are governed by *apparently* random and chaotic vorticity, energy dissipation, and near-fractal distribution of scales. Turbulence is still an unsolved problem in physics and actually a rather important one, since all flows in the real world become turbulent above a certain Reynolds number. However, not in all these cases turbulence governs the complete flow – structures are still distinguishable. This cannot be said about pure turbulence where equally important structures are equally distributed in the whole domain.

The topology of turbulent flows hardly yields insightful visualizations, but a statistical analysis of the topological structures can be applied successfully to measure the size of dissipation elements as shown by Wang and Peters [WP06]. Furthermore, Laney et al. [LBM$^+$06] statistically analyze topological structures to study a Rayleigh-Taylor instability.

8.4 Topological Complexity of 2D and 3D Vector Fields

Topological visualizations are limited to a rather moderate topological complexity which becomes manifest in the number of present topological features: if only very few features are present (or no features at all), a topological visualization fails. On the other hand, if there are too many topological features, their visualization fails as well due to clutter which is hard (or even impossible) to interpret.

This leads us to studying the topological complexity of vector fields in order to see where the limits for applying topological visualizations are. For the upper limit (i.e. the data is too complex), a number of technical and perceptional reasons are known. We show that there is an additional theoretical reason which strongly limits 3D topology to rather simple data sets. This reason lies in the fast growing number of sectors of different flow behavior. We show that – contrary to 2D vector fields – the number of sectors of different flow behavior grows in the worst case quadratically with the number of critical points.

8.4.1 Counting the Number of Sectors

We start with an analysis of existing 3D topological visualization approaches and consider the topological complexity of the treated data sets. Table 8.1 gives a collection of these techniques, including the topological complexity of the treated examples. We express this complexity by counting the number of critical points, boundary switch curves and separatrices which are present in the application. Table 8.1 does not intend to give an evaluation of the considered techniques because they focus on different data

[1]Note that a high number of features is not a problem in general. If they can be observed in a certain region of the data set only, they allow at least a distinction between this region and the rest of the domain. While this might not give intuitive feature-based visualizations, it can at least be used to steer other visualization techniques, see figure 7.23 on page 147. Furthermore, if a high amount of features is equally distributed but some kind of patterns are formed, this usually leads to revealing visualizations, see figure 4.13 on page 72.

reference	critical points	separatrices from critical points	boundary switch curves	separatrices from boundary switch curves
[HH91]	≈20	≈5	0	0
[GLL91]	≈2	0	0	0
[LDG98]	3	2	0	0
[MBS$^+$04]	0	0	≈10	≈10
[MBHJ03]	1	1	≈10	≈10
[GTS04b]	≈6	≈3	0	0
[SBSH04]	9	9	0	0
[TWHS03]	184	121	0	0
[WTHS04a]	184	121	13	22
[WTS$^+$05]	184	121	0	0
[TWHS07]	452	452	0	0

Table 8.1: 3D topological visualization approaches and their number of treated topological features. The last few rows of the table cite publications which represent the work of this thesis. Publications have only been listed if they are concerned with visualizing the topological structures.

sets or incorporate other visualization techniques as well. However, it reveals that most of the applications deal only with vector fields of a very low topological complexity. Only some papers [TWHS03, WTHS04a, WTS$^+$05, TWHS07] representing the work of this thesis consider fairly complex data sets, which was possible by using simplification concepts such as saddle connectors, i.e., classical topological methods were not appropriate there either.

We search for reasons why up to now topological methods have been applied only to rather simple data sets. Two classes of reasons are already known from the work on connectors (section 2.2.4):

1. *Technical reason:* 3D topological methods involve the integration of stream surfaces which is computationally more involved, less stable, and less accurate than the integration of stream lines in 2D.

2. *Perceptional reason:* The sectors of different flow behavior may have a complicated shape and hide each other, making a visual analysis of them a cumbersome task. Figure 8.1 shows an example of a vector field consisting of 4 saddles which create 6 sectors of different flow behavior. Even for this rather low number of sectors we observe the hiding effect making it hard to distinguish the different sectors.

In recent years, the first problem became more and more unimportant due to the dramatic increase of computing capacities and a number of new algorithmic solutions [Hul92, vW93, Gel01, SBM$^+$01]. One solution for the second problem is the connector approach as developed in the course of this thesis (see section 2.2.4).

Now we show that there is a third reason that topological methods are limited to low-complexity vector fields in 3D. We show that – simply spoken – the number of sectors of different flow behavior grows fast when the topological complexity of the vector field increases. As a measure of topological complexity, we take the number of present saddle points, since the separatrices emanate from there. Then we get a

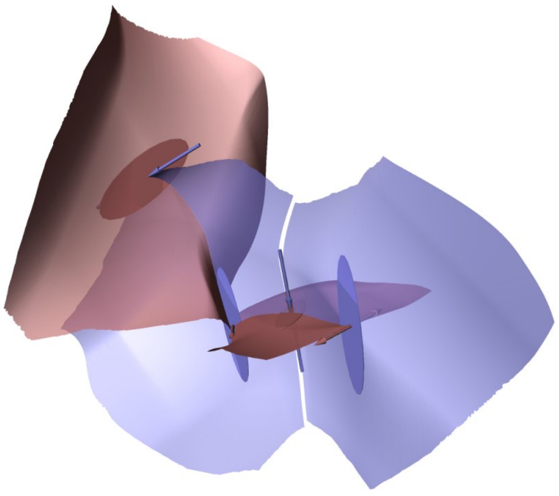


Figure 8.1: Simple topological skeleton consisting of 4 saddles. The 6 resulting sectors of different flow behavior can hardly be distinguished.

3. *Theoretical reason:* The number of sectors of different flow behavior grows in the worst case quadratically with the number of saddle points in a 3D vector field.

We show that this reason is a serious limitation of applying topological methods to 3D vector fields. To prove this reason, we present formulas to compute the number of sectors of different flow behavior. To see the difference, we do so for both 2D and 3D vector fields.

Sector Counting for 2D Vector Fields

2D vector fields generally consist of sources, sinks and saddles where a saddle creates 4 separation curves (see section 2.2). We get

Property 1 *Given a 2D vector field* $\mathbf{v}_{n_S}$ *consisting of* n_S *saddle points, the number* $sec(\mathbf{v}_{n_S})$ *of sectors of different flow behavior fulfills*

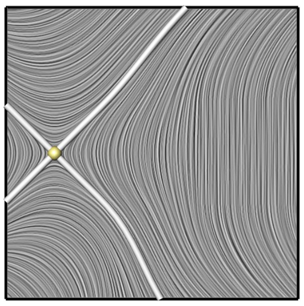
$$sec(\mathbf{v}_{n_S}) \leq 3n_s + 1 \tag{8.1}$$

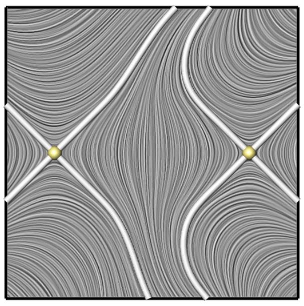
where the equality in (8.1) *can be reached.*

Property 1 essentially says that the number of sectors grows linearly with the number of saddle points. To show it, we start with a vector field consisting of only one saddle, as shown in figure 8.2a. This saddle divides the domain into four sectors. Now we insert a new saddle as shown in figure 8.2b. Since the different sectors are separated by stream lines which must not intersect each other, a new saddle replaces one of the old sectors by 4 new sectors, thus increasing the total number of sectors by 3. This gives

$$sec(\mathbf{v}_{n_S+1}) \leq sec(\mathbf{v}_{n_S}) + 3, \tag{8.2}$$

which is an inequality since separatrices may end in the same source/sink which reduces the number of sectors. Figure 8.2c illustrates this. (8.2) and $sec(\mathbf{v}_0) = 1$ gives (8.1). To complete the proof, we only have to show that the equality in (8.1) can be


(a) A single saddle point segments the domain into 4 sectors.


(b) An additionally included saddle increases the total number of sectors by 3.

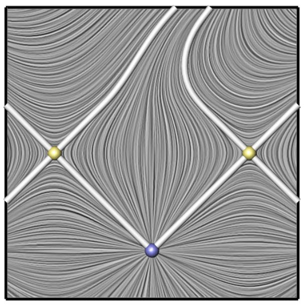

(c) If separatrices end in the same source/sink, two sectors are merged.

Figure 8.2: Sectors of different flow behavior in a 2D vector field.

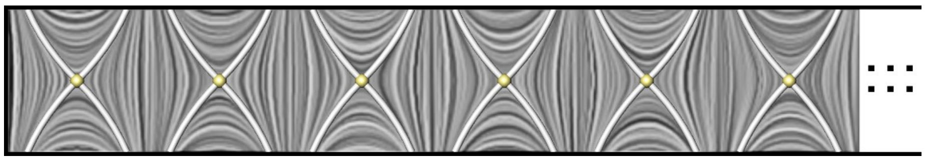


Figure 8.3: Constructed 2D vector field with maximal number of sectors, i.e., $\mathbf{v}_{n_S}$ with $sec(\mathbf{v}_{n_S}) = 3\,n_s + 1$.

reached. To do so, we construct a vector field $\mathbf{v}_{n_S}$ with $sec(\mathbf{v}_{n_S}) = 3\,n_s + 1$. Figure 8.3 illustrates the construction of such a simple vector field.

Property 1 gives a reason why 2D topological methods are rather popular even for fairly complex vector fields: the number of sectors to be distinguished grows only slowly (in fact linearly) with increasing topological complexity (i.e., the number of saddle points). As we will show now, this does not hold for 3D vector fields.

Sector Counting for 3D Vector Fields

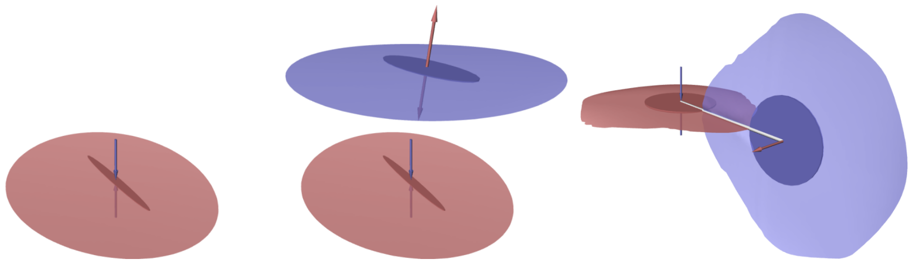
For 3D vector fields, the number of sectors of different flow behavior depends in the worst case quadratically on the number of saddle points. We formulate

Property 2 *Given a 3D vector field $\mathbf{v}_{n_R,\,n_A}$ consisting of n_R repelling saddles and n_A attracting saddles, for the number $sec(\mathbf{v}_{n_R,\,n_A})$ of sectors of different flow behavior the inequality holds*

$$sec(\mathbf{v}_{n_R,\,n_A}) \;\leq\; (n_R+1)(n_A+1) \tag{8.3}$$

where the equality in (8.3) *can be reached.*

To show property 2, we start with a simple vector field $\mathbf{v}_{1,0}$ consisting only of one repelling saddle $\mathbf{x}_R$, as shown in figure 8.4a. The separation surface created by $\mathbf{x}_R$ divides $\mathbf{v}_{1,0}$ into two sectors. If we insert a new saddle, this can be either an attracting saddle $\mathbf{y}_A$ or a repelling saddle $\mathbf{y}_R$ as well. In the last case, the separation surfaces of $\mathbf{x}_R$ and $\mathbf{y}_R$ create three sectors of different flow behavior since they must not intersect. In case of a newly inserted attracting saddle $\mathbf{y}_A$, two cases are possible:



(a) Simple vector field with one $\mathbf{x}_R$: two sectors are present.

(b) Inserting $\mathbf{y}_A$ *without* connector gives 3 sectors.

(c) Inserting $\mathbf{y}_A$ *with* connector gives 4 sectors.

Figure 8.4: Correlation between number of sectors, saddles and connectors.

- The separation surface of $\mathbf{y}_A$ does not intersect the separation surface of $\mathbf{x}_R$. In this case, one of the old sectors is divided into two new sectors, and the total number of sectors is increased by 1. Figure 8.4b gives an illustration.
- The separation surface of $\mathbf{y}_A$ intersects the separation surface of $\mathbf{x}_R$. In this case, each of the two old sectors is divided into two new sectors. Thus, the total number of sectors is increased by 2. Figure 8.4c illustrates this.

As we can see from the simple example above, the total number of sectors does not only depend on the number of saddles but also on the number of saddle connectors.

For now we assume that every repelling saddle has a connector to every attracting saddle. Given a vector field $\mathbf{v}_{n_R,\,n_A}$, we consider the insertion of a new attracting saddle $\mathbf{x}_A$. Assuming that $\mathbf{x}_A$ has a connector to all n_R repelling saddles of $\mathbf{v}_{n_R,\,n_A}$, $\mathbf{x}_A$ divides n_R+1 of the old sectors into two new sectors each. We get

$$sec(\mathbf{v}_{n_R,\,n_A+1}) \;\leq\; sec(\mathbf{v}_{n_R,\,n_A}) + n_R + 1 \tag{8.4}$$

and in a similar way

$$sec(\mathbf{v}_{n_R+1,\,n_A}) \;\leq\; sec(\mathbf{v}_{n_R,\,n_A}) + n_A + 1. \tag{8.5}$$

(8.4), (8.5) and $sec(\mathbf{v}_{0,0}) = 1$ give (8.3).

To complete the proof of property 2, we construct an example vector field $\mathbf{v}_{n_R,\,n_A+1}$ with $sec(\mathbf{v}_{n_R,\,n_A}) = (n_R+1)(n_A+1)$. To do so, we use the topological vector field construction approach described in section 7.1. We place the n_R repelling saddles to the locations $(1,\frac{n_A}{2},-d)$, $(2,\frac{n_A}{2},-d)$, $\ldots$, $(n_R,\frac{n_A}{2},-d)$ in such a way that the inflow plane of each saddle is parallel to the $y-z$ plane of the underlying Euclidean coordinate system. Furthermore, we place n_R+1 sources at the locations $(0.5,\frac{n_A}{2},-d)$, $(1.5,\frac{n_A}{2},-d)$, ..., $(n_R+0.5,\frac{n_A}{2},-d)$. To place the n_A attracting saddles, we choose the locations $(\frac{n_R}{2},1,d)$, $(\frac{n_R}{2},2,d)$, ..., $(\frac{n_R}{2},n_A,d)$. In addition we place n_A+1 sinks at the locations $(\frac{n_R}{2},0.5,d)$, $(\frac{n_R}{2},1.5,d)$, ..., $(\frac{n_R}{2},n_A+0.5,d)$. The positive number d describes the distance of the two rows of saddles. Figure 8.5a illustrates the location of the critical points for the example $\mathbf{v}_{4,4}$.

In the next step we have to construct a system of connectors such that each repelling saddle is connected to each attracting saddle. This is a set of $n_R \cdot n_A$ curves which must not intersect each other (except in the saddles themselves). Given the arrangement of critical points described above, this can easily be done as illustrated in figure 8.5a for $\mathbf{v}_{4,4}$. The complete constructed vector field ensures that for any source $\mathbf{x}_{So}$ and for any

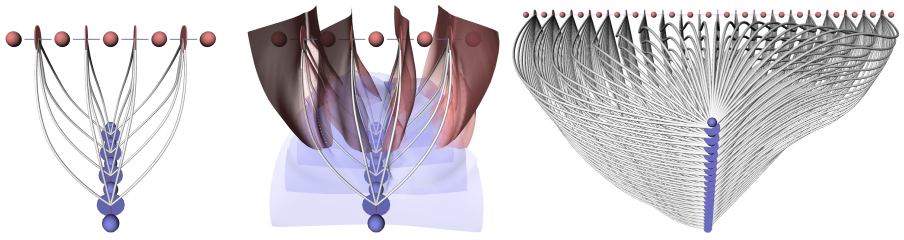

(a) $\mathbf{v}_{4,4}$: critical points and 16 saddle connectors.
(b) $\mathbf{v}_{4,4}$: separation surfaces.
(c) $\mathbf{v}_{20,20}$: critical points and 400 saddle connectors.

Figure 8.5: Constructed 3D vector fields $\mathbf{v}_{4,4}$ and $\mathbf{v}_{20,20}$ with maximal number of sectors.

sink $\mathbf{x}_{Si}$ there are stream lines starting in $\mathbf{x}_{So}$ and ending in $\mathbf{x}_{Si}$. Figure 8.5b shows the complete topological skeleton of $\mathbf{v}_{4,4}$. Figure 8.5c shows an example of the constructed vector field $\mathbf{v}_{20,20}$ consisting of 441 sectors of different flow behavior.

Property 2 can be concretized by incorporating not only the number of critical points but also the number of connectors:

Property 3 *Given a 3D vector field $\mathbf{v}_{n_R,\,n_A,\,n_{Co}}$ consisting of n_R repelling saddles, n_A attracting saddles and n_{Co} saddle connectors, for the number $sec(\mathbf{v}_{n_R,\,n_A,\,n_{Co}})$ of sectors of different flow behavior the inequality holds*

$$sec(\mathbf{v}_{n_R,\,n_A,\,n_{Co}}) \;\leq\; n_R + n_A + n_{Co} + 1. \tag{8.6}$$

To show property 3, we assume a vector field $\mathbf{v}_{n_R,\,n_A,\,n_{Co}}$ in which we insert a new attracting saddle $\mathbf{x}_A$. Further we assume that $\mathbf{x}_A$ creates m new saddle connectors, i.e. $\mathbf{x}_A$ is connected with m repelling saddles of $\mathbf{v}_{n_R,\,n_A,\,n_{Co}}$. In this case, $m+1$ sectors of the old vector field are divided into two new sectors each. We obtain

$$sec(\mathbf{v}_{n_R,\,n_A+1,\,n_{Co}+m}) \;\leq\; sec(\mathbf{v}_{n_R,\,n_A,\,n_{Co}}) + m + 1 \tag{8.7}$$

and in a similar way

$$sec(\mathbf{v}_{n_R+1,\,n_A,\,n_{Co}+m}) \;\leq\; sec(\mathbf{v}_{n_R,\,n_A,\,n_{Co}}) + m + 1. \tag{8.8}$$

This and $sec(\mathbf{v}_{0,0,0}) = 1$ gives (8.6).

Remarks

1. The conditions in properties 2 and 3 are formulated as inequality because - similar to the 2D case - separatrices might end in the same critical points which leads to a reduction of the total number of sectors.

2. Properties 2 and 3 did not consider separatrices emanating from boundary switch curves. However, their quantitative behavior is similar to the separatrices from saddle connectors: the number of sectors grows in the worst case quadratically to the number of boundary switch curves.

3. Property 3 shows that in the best case the number of sectors grows linearly with the number of saddles. This happens if no saddle connectors exist at all. However, the examples throughout this thesis have shown that a higher number of saddle connectors usually exists.

4. Properties 2 and 3 considered at most one connector between a repelling saddle $\mathbf{x}_R$ and an attracting saddle $\mathbf{x}_A$. If multiple connectors are present, a sector (describing the flow from one particular source to one particular sink) may consist of different unconnected parts.

5. The sector counting presented here is a worst case estimation. Although an average case estimation would be useful, we are not aware of any approaches for this.

8.4.2 Interpretation of Sector Counting

From the sector counting approach in the previous section we draw the conclusion that classical 3D topological methods are limited to topologically rather simple vector fields. If the topological complexity (i.e. the number of saddles) grows, the number of sectors of different flow behavior very soon exceeds the limit of what can be distinguished in visualization. The following solutions can be applied:

- The topological skeleton may be simplified by removing unimportant critical points or collapse clusters of critical points to a higher order one. These methods are well-established for 2D vector fields [dLvL99a, TSH01a]. For 3D vector fields, there is only the solution described in chapter 6, which allows to replace a cluster of critical points with a higher order one, but does not take separatrices into account.

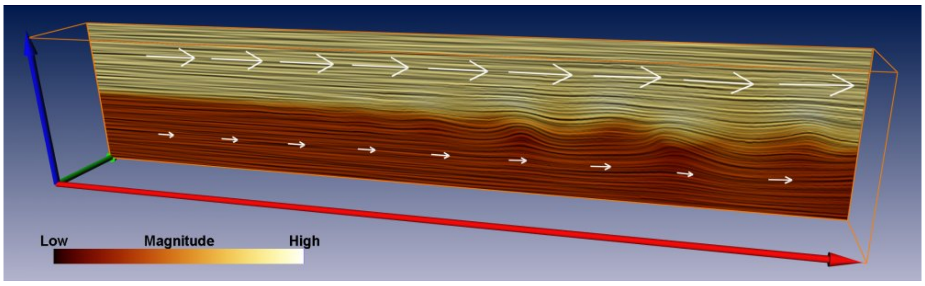
- The visualization itself can be simplified by depicting only parts of the skeleton. For example, one might display connectors instead of separation surfaces.

- Even if the complexity is too rich for a direct topological visualization, topological information can be used to parameterize other visualization techniques as shown in section 7.3.

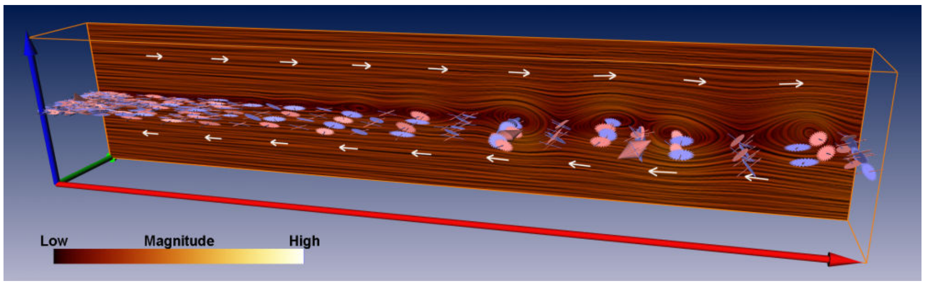
While dealing with a variety of data sets, we encountered vector fields where topological methods totally fail due to the absence of critical points and boundary switch curves. While it might not be possible to overcome this problem for all kinds of data, there is a solution for an important class of flow fields that exhibit a constant ambient flow part: all convections, i.e., coherent structures, move with nearly the same velocity and direction inside the flow. Their corresponding topological structures cannot be extracted since the ambient flow part cancels out the critical points. This clearly shows the Galilean-variance of topological examinations if applied to the velocity field. By subtracting the ambient flow part, e.g. choosing a certain frame of reference, the coherent structures become visible using topological methods. To ensure meaningful results, this manipulation must be motivated by the physical interpretation of the data. A scheme for computing the ambient part of a flow has been presented by Wiebel et al. [WGS05].

Consider the mixing layer visualized in figure 8.6a, where the flow moves downstream in both layers and the magnitude of the upper layer is three times larger than in the lower layer. The data set has been computed with a pseudo-spectral direct numerical simulation employing the computational domain and boundary conditions of Comte, Silvestrini & Bégou [CSB98]. The Reynolds number is 100 based on the initial shear-layer thickness and convection velocity.

No critical points are present in this original frame of reference. In figure 8.6b we have chosen to subtract the constant vector field $(1,0,0)^T$. This yields the frame



(a) Original frame of reference. No critical points are present.



(b) Frame of reference chosen such that both layers have the same magnitude. 348 saddle points have been detected.

Figure 8.6: Mixing layer. Topological analysis made possible by removing the ambient part of the flow.

of reference where the flow in both layers has the same magnitude, but a different direction. The physical interpretation behind this manipulation is that we move as a observer with the same velocity and direction as the convections. The topology of this frame clearly shows formations of focus saddles indicating Kelvin-Helmholtz vortices, which alternate with formations of node saddles.

This example shows that topological methods yield expressive visualizations even for initially topologically sparse vector fields if a frame of reference can be chosen with regards to physical interpretation.

Chapter 9

Conclusion and Future Work

This thesis presented a number of algorithms for the extraction, tracking, and visualization of topological structures of vector fields.

The new topological concepts of *saddle connectors* and *boundary switch connectors* allow for a visually simplified representation of the topological skeletons of complex 3D vector fields. Connectors yield for the first time expressive visualizations of complex topological skeletons with a higher number of critical points, boundary switch curves and separatrices. They can be interpreted as a "skeleton of a skeleton" concept. In fact, a geometrically simplified representation is computed for the system of separation surfaces. Although we have shown that this is a useful compromise between the amount of coded information and the expressiveness of the visualization for complex topological skeletons, one important property of the topological skeleton is lost: the unique partition of the vector field into areas of similar flow behavior. In fact, using connectors it is no longer possible to uniquely infer the flow behavior of $\mathbf{v}$ from any point of the domain. Nevertheless, by representing the connectors using double flow ribbons the user can comprehend the topology of the 3D vector field as well as the flow behavior in particular points of the domain. Furthermore, with the possibility to interactively demand the display of single separation surfaces, the user can explore the partitioning of the whole domain.

Pioneering work has been done on the field of *higher order topology* of 3D vector fields. We introduced a classification of 3D higher order critical points to the Computer Graphics and Visualization community and presented the first visualization technique for them. Based on this theory, two novel applications have been developed: *simplification* and *construction* of 3D vector fields. The simplification method is based on a novel algorithm for extracting higher order critical points and allows to replace the topological skeleton of a cluster of first order critical points by a single higher order icon. There is room for future work since certain types of flow with swirling behavior cannot be handled, and separatrices are not considered by the simplification method. The construction technique allows to automatically create a piecewise linear vector field from a previously modeled topological skeleton. This research has the potential of being the starting point for a number of applications like topology preserving compression, optimizing flow, and topological simplification incorporating separatrices. On a more theoretical level, there is still the open question on how the index of a 3D higher order critical point relates to the number of sectors of different flow behavior.

We developed the first generic approach to feature extraction treating local and global features: the *Unified Feature Extraction Architecture* (UFEA) consists of three core algorithms only, but allows to extract and track a rich variety of geometrically defined, local and global features evolving in scalar and vector fields. Complex feature extraction techniques are built as a combination of the three basic algorithms. By using generic concepts and grid independent algorithms, this software architecture aims at a broad applicability while alleviating the implementational expenses. There are two major concepts behind UFEA: the *Connectors approach* provides the means to treating global features such as closed stream lines and saddle connectors. The concept of *Feature Flow Fields* (FFF) has been greatly extended in this thesis. We have shown how every problem defined by the Parallel Vectors operator [PR99] can be reformulated using FFF, which gives an alternative extraction scheme for a number of features. Furthermore, we presented solutions to make every FFF-based extraction technique compatible to out-of-core data handling.

Based on UFEA, we have shown how to extract the topology of steady vector fields and track it in time-dependent vector fields. The work included a new approach to detecting *closed stream lines* in steady 2D vector fields, which – in contrast to previous approaches – does not depend on any underlying grid structure of the vector field. This has been the foundation of a novel method for tracking closed stream lines in 2D time-dependent vector fields, which is robust against *cyclic fold bifurcations*. Other global bifurcations like *saddle connections* and *periodic blue sky bifurcations* were treated as well. Future research clearly goes into the direction of detecting closed structures in 3D vector fields. The extension of our approach to 3D vector fields seems to be possible but is not straightforward, since in this case 3D stream-hypersurfaces of 4D vector fields have to be intersected.

We introduced a topology-based visualization approach for *two-parameter-dependent* 2D vector fields. Based on UFEA, we gave an algorithm to tracking fold bifurcations and Hopf bifurcations in the 4D domain and discussed local bifurcations introduced by the additional parameter: fold-fold and Hopf-fold bifurcations. We believe that data sets depending on a number of additional parameters will be common in a not too distant future. The most challenging issue for future research is the extension of our approach from 2D to 3D two-parameter-dependent vector fields. While most of the topological concepts extend straightforwardly, the main problem seems to be an adequate visual representation of the resulting 5D features. Furthermore, we concentrated on the evolution of the critical points – tracking of other topological structures in two-parameter-dependent vector fields is left for future research.

We utilized the technology behind UFEA not only for topological examinations, but applied it also to assess vortices. We presented a new algorithm to *extract and track* core lines of swirling motion in 3D time-dependent flows. For the first time, we addressed the *swirling motion of path lines* in unsteady flows – leading to a novel mathematical characterization of swirling particle cores. This work has been set into relation to previous work by formulating a *unified notation* of swirling motion in 2D and 3D flows. Our method for path lines is a generalization of the approach by Sujudi/Haimes [SH95b] and clearly inherits its limitations. In particular, the method can result in false positives, i.e., lines that actually lie off the desired core line. Also when noise is present the method may extract a variety of short lines. Both issues can be treated by filtering the output by length and certain angle criteria. Roth and Peikert pointed out in [RP98] that the method of Sujudi/Haimes has its limitations in settings where curved boundaries are involved. Our method might have comparable limitations and it is an interesting point for further studies to see if the higher-order methods in [RP98] can be extended to path lines in an analogous way. Cores of swirling motion have an intuitive interpretation, accompany the behavior of integral curves and their extraction is comparatively simple and fast.

However, different vortex definitions exist and swirling motion is only one way to assess vortices. In fact, most vortex definitions given in the fluid dynamics community are either Galilean invariant or even objective (see [Hal05, Hun87, JH95]), but all types of swirling motion cores are not invariant under such transformations. Therefore, we introduced the notion of *Galilean invariant vortex core lines* as extremum lines of Galilean invariant scalar quantities like pressure, Q, or λ_2. Extremum lines allow parameter free extraction (as opposed to isosurfaces or volume rendering) as well as a quantitative analysis, and they correctly identify location and extent of the vortices. We developed an extraction technique based on height ridges with the drawback of requiring second order derivatives. A second extraction technique – requiring first order derivatives only – links the areas of topology and vortex analysis by showing how the core of a vortex can be defined as a separatrix of a topological skeleton of a derived vector field. This opens opportunities for future work where topological tools, e.g. simplification, can be applied to the field of vortex analysis. Again, the extraction methods for vortex core lines are based on UFEA, too.

We discussed a number of aspects like the topology of path lines or the sensitivity of topological methods to noise. Furthermore, we studied their applicability for creating expressive, feature revealing visualizations by estimating the *topological complexity* of vector fields. It turns out that simplification methods for 3D vector fields have to be a major future research direction to allow topology-based visualizations even for very complex 3D data sets. This thesis already pushed the limit by developing the concept of connectors and the first topological 3D simplification technique, but not all problems have been solved fundamentally yet. As resolution and complexity of the data sets have evolved significantly in the last years, the challenge of understanding the intricate structures has made automatic feature extraction necessary. Using a well-defined set of techniques, features are a way of reducing the sheer amount of information to a specifiable degree. And as shown in this thesis, a variety of features can be defined and handled in a mathematically profound manner by means of topology.

Bibliography

[AS92] L. Abraham and K. Shaw. *Dynamics, The Geometry of Behaviour*. Addison-Wesley, 1992.

[Asi93] D. Asimov. Notes on the topology of vector fields and flows. Technical report, NASA Ames Research Center, 1993. RNR-93-003.

[Bad90] R. Bader. *Atoms in Molecules - A Quantum Theory*. Oxford University Press, Oxford, 1990.

[Bak91] P. G. Bakker. *Bifurcations in Flow Patterns (Theory and Applications of Transport in Porous Media)*. Kluwer Academic Publishers, 1991.

[BD86] P. Bonckaert and F. Dumortier. Smooth invariant curves for germs of vector fields in $\mathbb{R}^3$ whose linear part generates rotations. *J. Diff. Eqns.*, 62:95–116, 1986.

[BH99] R. Batra and L. Hesselink. Feature comparisons of 3-D vector fields using earth mover's distance. In *Proc. IEEE Visualization '99*, pages 105–114, 1999.

[BKH99] R. Batra, K. Kling, and L. Hesselink. Topology based vector field comparison using graph methods. In *Proc. IEEE Visualization '99, Late Breaking Hot Topics*, pages 25–28, 1999.

[BML95] B. G. Becker, N. L. Max, and D. A. Lane. Unsteady flow volumes. In *Proc. IEEE Visualization 1995*, pages 19–24, 1995.

[BP96] E. Boring and A. Pang. Directional flow visualization of vector fields. In *Proc. IEEE Visualization 1996*, pages 389–392, 1996.

[BP02] D. Bauer and R. Peikert. Vortex tracking in scale space. In *Data Visualization 2002. Proc. VisSym 02*, pages 233–240, 2002.

[BR63] R. L. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *Proc. of the IEEE Fall Joint Computer Conference*, pages 445–458, 1963.

[BS95] D. Banks and B. Singer. A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):151–163, 1995.

[BSH96] H. Battke, D. Stalling, and H.-C. Hege. *Visualization and Mathematics 1997*. Springer, Heidelberg, 1996.

[CCL03] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the morse-smale complex and the connolly function. In *SCG: Proc. of the 19. Annual Symposium on Computational Geometry*, pages 351–360, 2003.

[CL93] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. *Computer Graphics*, 27:263–272, 1993.

[CMBC93] R. Crawfis, N. Max, B. Becker, and B. Cabral. Volume rendering of 3d scalar and vector fields at LLNL. In *Proc. Supercomputing 1993*, pages 570–576, 1993.

[CML$^+$07] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):769–785, July - August 2007.

[CPC90] M. S. Chong, A. E. Perry, and B. J. Cantwell. A general classification of three-dimensional flow fields. *Physics of Fluids A*, 2(5):765–777, 1990.

[CS03] H. Carr and J. Snoeyink. Path seeds and flexible isosurfaces - using topology for exploratory visualization. In *Data Visualization 2003. Proc. VisSym 03*, pages 49–58, 2003.

[CSB98] P. Comte, J. Silvestrini, and P. Bégou. Streamwise vortices in Large-Eddy Simulations of mixing layer. *Eur. J. Mech. B*, 17:615–637, 1998.

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

[DI98] F. Dumortier and S. Ibáñez. Singularities of vector fields on $\mathbb{R}^3$. *J. Nonlinearity.*, 11:1037–1047, 1998.

[DLBB07] S. Depardon, J. Lasserre, L. Brizzi, and J. Borée. Automated topology classification method for instantaneous velocity fields. *Experiments in Fluids*, 42(5):697–710, May 2007.

[dLvL98] W. de Leeuw and R. van Liere. Comparing LIC and spot noise. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 359–365, Los Alamitos, 1998. IEEE Computer Society Press.

[dLvL99a] W. de Leeuw and R. van Liere. Collapsing flow topology using area metrics. In *Proc. IEEE Visualization '99*, pages 149–354, 1999.

[dLvL99b] W. de Leeuw and R. van Liere. Visualization of global flow structures using multiple levels of topology. In *Data Visualization 1999. Proc. VisSym 99*, pages 45–52, 1999.

[dLvW95] W. C. de Leeuw and J. van Wijk. Enhanced spot noise for vector field visualization. In *Proc. IEEE Visualization 1995*, pages 233–239, 1995.

[Ebe96] D. Eberly. *Ridges in Image and Data Analysis*. Kluwer Acadamic Publishers, Dordrecht, 1996.

[EGM$^+$94] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. *Journal of Mathematical Imaging and Vision*, 4(4):353–373, 1994.

[EHNP03] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-smale complexes for piecewise linear 3-manifolds. In *Proc. 19th Sympos. Comput. Geom. 2003*, pages 361 – 370, 2003.

[ES03] J. Ebling and G. Scheuermann. Clifford convolution and pattern matching on vector fields. In *Proc. IEEE Visualization 2003*, pages 193–200, 2003.

[EYSK94] D. S. Ebert, R. Yagel, J. Scott, and Y. Kurzion. Volume rendering methods for computational fluid dynamics visualization. In *Proc. IEEE Visualization 1994*, pages 232–239, 1994.

[FG82] P. Firby and C. Gardiner. *Surface Topology*, chapter 7, pages 115–135. Ellis Horwood Ltd., 1982. Vector Fields on Surfaces.

[For98] R. Forman. Morse theory for cell-complexes. *Advances in Mathematics*, 134(1):90–145, 1998.

[For02] R. Forman. A user's guide to discrete morse theory. Séminaire Lotharingien de Combinatoire, B48c, 2002.

[Gel01] A. V. Gelder. Stream surface generation for fluid flow solutions on curvilinear grids. In *Data Visualization 2001. Proc. VisSym 01*, 2001.

[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 (Proceedings Visualization 2007)*, 13(6), November - December 2007.

[GH86] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer, 2nd edition, 1986.

[GLL91] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *Proc. IEEE Visualization '91*, pages 33–40, 1991.

[GLT$^+$07] C. Garth, R. S. Laramee, X. Tricoche, J. Schneider, and H. Hagen. Extraction and visualization of swirl and tumble motion from engine simulation data. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 121–135. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.

[GNP$^+$05] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3d scalar fields. In *Proc. IEEE Visualization 2005*, pages 535–542, 2005.

[GNPH07] A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of morse-smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2007)*, 13(6), November - December 2007.

[Got90] D. H. Gottlieb. Vector fields and classical theorems of topology. Rendiconti del Seminario Matematico e Fisico, Milano, 60, 1990.

[Got96] D. H. Gottlieb. All the way with gauss-bonnet and the sociology of mathematics. *The American Mathematical Monthly*, 103(6):457–469, June - July 1996.

[GTP$^+$07] B. Günther, F. Thiele, R. Petz, W. Nitsche, J. Sahner, T. Weinkauf, and H.-C. Hege. Control of separation on the flap of a three-element high-lift configuration. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, U.S.A., January 2007.

[GTS$^+$04a] C. Garth, X. Tricoche, T. Salzbrunn, T. Bobach, and G. Scheuermann. Surface techniques for vortex visualization. In *Proc. IEEE TCVG Symposium on Visualization 2004*, pages 155–164, 2004.

[GTS04b] C. Garth, X. Tricoche, and G. Scheuermann. Tracking of vector field singularities in unstructured 3D time-dependent datasets. In *Proc. IEEE Visualization 2004*, pages 329–336, 2004.

[Hai99] R. Haimes. Using residence time for the extraction of recirculation regions. AIAA Paper 99-3291, 1999.

[Hal00] G. Haller. Finding finite-time invariant manifolds in two-dimensional velocity fields. *Chaos*, 10(1):99–108, 2000.

[Hal01] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D*, 149:248–277, 2001.

[Hal04] G. Haller. Exact theory of unsteady separation for two-dimensional flows. *Journal of Fluid Mechanics*, 512:257–311, Aug. 2004.

[Hal05] G. Haller. An objective definition of a vortex. *J. Fluid Mech.*, 525:1–26, 2005.

[HEWK03] E. Heiberg, T. Ebbers, L. Wigström, and M. Karlsson. Three dimensional flow characterization using vector pattern matching. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):313–319, 2003.

[HG00] H. Hauser and E. Gröller. Thorough insights by enhanced visualization of flow topology. In *9th international symposium on flow visualization*, 2000.

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

[HH91] J. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11:36–46, May 1991.

[HLD07] H. Hauser, R. S. Laramee, and H. Doleisch. Topology-based versus feature-based flow analysis - challenges and an application. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 79–90. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.

[HSSZ97] H.-C. Hege, M. Seebass, D. Stalling, and M. Zöckler. A generalized marching cubes algorithm based on non-binary classifications. *Preprint SC-97-05*, 1997. ZIB, Berlin.

[Hul92] J. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proc. IEEE Visualization '92*, pages 171–177, 1992.

[Hun87] J. Hunt. Vorticity and vortex dynamics in complex turbulent flows. *Proc CANCAM, Trans. Can. Soc. Mec. Engrs*, 11:21, 1987.

[HWPH04] H.-C. Hege, T. Weinkauf, S. Prohaska, and A. Hutanu. Distributed visualization and analysis of fluid dynamics data. In *Proc. Fourth International Symposium on Advanced Fluid Information and Transdisciplinary Fluid Integration*, Sendai, Japan, November 2004.

[HWPH05] H.-C. Hege, T. Weinkauf, S. Prohaska, and A. Hutanu. Towards distributed visualization and analysis of large flow data. *JSME International Journal Series B*, 48(2):241–246, May 2005.

[HWSE99] W. Heidrich, R. Westermann, H.-P. Seidel, and T. Ertl. Applications of pixel textures in visualization and realistic image synthesis. In *Proc. Symposium on Interactive 3D Graphics '99*, pages 127–134, 1999.

[IG97] V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 421–425, 1997.

[JEH02] B. Jobard, G. Erlebacher, and M. Y. Hussaini. Lagrangian-eulerian advection of noise and dye textures for unsteady flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 08(3):211–222, 2002.

[JH95] J. Jeong and F. Hussain. On the identification of a vortex. *J. Fluid Mechanics*, 285:69–94, 1995.

[JL97] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proceedings 8th Eurographics Workshop on Visualization in Scientific Computing*, pages 57–66, Boulogne, 1997.

[JMT02] M. Jiang, R. Machiraju, and D. Thompson. Geometric verification of swirling features in flow fields. In *Proc. IEEE Visualization 2002*, pages 307–314, 2002.

[KE07] T. Klein and T. Ertl. Scale-space tracking of critical points in 3d vector fields. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 35–50. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.

[KH91] R. V. Klassen and S. J. Harrington. Shadowed hedgehogs: A technique for visualizing 2D slices of 3D vector fields. In *Proc. IEEE Visualization 1991*, pages 148–153, 1991.

[KHL99] D. Kenwright, C. Henze, and C. Levit. Feature extraction of separation and attachment lines. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):135–144, 1999.

[KJ00] H.-J. Kaltenbach and G. Janke. Direct numerical simulation of flow separation behind a swept, rearward-facing step at re$_H$=3000. *Physics of Fluids*, 12:2320–2337, 2000.

[KKKW05] J. Krüger, P. Kipfer, P. Kondratieva, and R. Westermann. A particle system for interactive visualization of 3D flows. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):744–756, 2005.

[KL96] D. N. Kenwright and D. A. Lane. Interactive time-dependent particle tracing using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):120–129, 1996.

[KML99] R. M. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing multivalued data from 2D incompressible flows using concepts from painting. In *Proc. IEEE Visualization 1999*, pages 333–340, 1999.

[KOD$^+$05] B. Krauskopf, H. M. Osinga, E. J. Doedel, M. E. Henderson, J. M. Guckenheimer, A. Vladimirsky, M. Dellnitz, and O. Junge. A survey of methods for computing (un)stable manifolds of vector fields. *Int. J. Bifurcation & Chaos*, 15(3):763–791, 2005.

[KS91] D. Kalivas and A. Sawchuk. A region matching motion estimation algorithm. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 54(2):275–288, Sept. 1991.

[KvD93] J. J. Koenderinck and A. J. van Doorn. Local features of smooth shapes: ridges and courses. In B. C. Vemuri, editor, *Proc. SPIE Int. Soc. Opt. Eng.*, volume 2031, pages 2–13, June 1993.

[LBH98] Y. Lavin, R. Batra, and L. Hesselink. Feature comparisons of vector fields using earth mover's distance. In *Proc. IEEE Visualization '98*, pages 103–109, 1998.

[LBM$^+$06] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2006)*, 12(5):1053–1060, September - October 2006.

[LDG98] H. Löffelmann, H. Doleisch, and E. Gröller. Visualizing dynamical systems near critical points. In *Spring Conference on Computer Graphics and its Applications*, pages 175–184, Budmerice, Slovakia, 1998.

[LFR03] S. Lodha, N. Faaland, and J. Renteria. Topology preserving top-down compression of 2d vector fields using bintree and triangular quadtrees. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):433–442, 2003.

[LGD$^+$05] R. S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser, and H. Hagen. Visual analysis and exploration of fluid flow in a cooling jacket. In *Proc. IEEE Visualization 2005*, pages 623–630, 2005.

[LHD$^+$04] R. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. Post, and D. Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum*, 23(2):143–161, 2004.

[LHZP07] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post. Topology-based flow visualization, the state of the art. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 1–19. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.

[LJH03] R. S. Laramee, B. Jobard, and H. Hauser. Image space based visualization of unsteady flow on surfaces. In *Proc. IEEE Visualization 2003*, pages 131–138, 2003.

[LMG97] H. Löffelmann, L. Mroz, and E. Gröller. Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems. *The Visual Computer*, 13(8):359–369, 1997.

[LPSW96] S. K. Lodha, A. Pang, R. E. Sheehan, and C. M. Wittenbrink. Uflow: Visualizing uncertainty in fluid flow. In *Proc. IEEE Visualization 1996*, pages 249–254, 1996.

[LRR00] S. Lodha, J. Renteria, and K. Roskin. Topology preserving compression of 2D vector fields. In *Proc. IEEE Visualization 2000*, pages 343–350, 2000.

[LVRL06] W.-C. Li, B. Vallet, N. Ray, and B. Levy. Representing higher-order singularities in vector fields on piecewise linear surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1315–1322, 2006. Proceedings Visualization '06.

[MAD05] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *Proc. IEEE Visualization 2005*, pages 479–486, 2005.

[MB95] N. Max and B. Becker. Flow visualization using moving textures. In *Proceedings of the ICASW/LaRC Symposium on Visualizing Time-Varying Data*, pages 77–87, 1995.

[MBC93] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proc. Visualization 93*, pages 19–24, 1993.

[MBHJ03] K. Mahrous, J. Bennett, B. Hamann, and K. Joy. Improving topological segmentation of three-dimensional vector fields. In *Data Visualization 2003. Proc. VisSym 03*, pages 203–212, 2003.

[MBS$^+$04] K. Mahrous, J. Bennett, G. Scheuermann, B. Hamann, and K. Joy. Topological segmentation in three-dimensional vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):198–205, 2004.

[MK97] H. Miura and S. Kida. Identification of tubular vortices in turbulence. *J. Physical Society of Japan*, 66(5):1331–1334, 1997.

[MKFI97] X. Mao, M. Kikukawa, N. Fujeta, and N. Imamiya. Line integral convolution for 3d surfaces. In W. Lefer and M. Grave, editors, *Visualization in Scientific Computing '97*, pages 57–69. Springer, Berlin, 1997.

[MPSS05] O. Mallo, R. Peikert, C. Sigg, and F. Sadlo. Illuminated lines revisited. In *Proc. IEEE Visualization 2005*, pages 19–26, 2005.

[MR02] S. Mann and A. Rockwood. Computing singularities of 3D vector fields with geometric algebra. In *Proc. IEEE Visualization 2002*, pages 283–289, 2002.

[MRV96] K.-L. Ma, J. V. Rosendale, and W. Vermeer. 3d shock wave visualization on unstructured grids. In *VVS '96: Proceedings of the 1996 symposium on Volume visualization*, pages 87–94, 1996.

[MSDS] H. Meuer, E. Strohmaier, J. Dongarra, and H. Simon. Top500 supercomputer sites. http://www.top500.org/.

[MW07] N. Max and T. Weinkauf. Critical points of the electric field from a collection of point charges. In *Proc. Topo-In-Vis 2007*, Grimma, Germany, March 2007.

[NAS90] NASA Langley Research Center. Wake vortex study at wallops island, May 1990. http://nix.larc.nasa.gov/info?id=EL-1996-00130&orgid=1.

[NE94] B. Noack and H. Eckelmann. A low-dimensional Galerkin method for the three-dimensional flow around a circular cylinder. *Phys. Fluids*, 6:124–143, 1994.

[Nie97] G. Nielson. Tools for computing tangent curves and topological graphs for visualizing piecewise linearly varying vector fields over triangulated domains. In G. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization*, pages 527–562. IEEE Computer Society, 1997.

[PCM03] V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38(2):249–268, 2003.

[PFTV91] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Comuting*. Cambridge University Press, Cambridge, 1991.

[Pow64] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[PR99] R. Peikert and M. Roth. The parallel vectors operator - a vector field visualization primitive. In *Proc. IEEE Visualization 99*, pages 263–270, 1999.

[PS93] H.-G. Pagendarm and B. Seitz. An algorithm for detection and visualization of discontinuities in scientific data fields applied to flow data with shock waves. In P. Palamidese, editor, *Scientific Visualization: Advanced Software Techniques*, pages 161–177. Ellis Horwood Limited, 1993.

[PS07] R. Peikert and F. Sadlo. Topology-guided visualization of constrained vector fields. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 21–34. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.

[PVH$^+$02] F. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch. Feature extraction and visualisation of flow fields. In *Proc. Eurographics 2002, State of the Art Reports*, pages 69–100, 2002.

[PVH$^+$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):775–792, 2003.

[PWL97] A. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, November 1997.

[PWS$^+$06] C. Petz, T. Weinkauf, H. Streckwall, F. Salvatore, B. Noack, and H.-C. Hege. Vortex structures at a rotating ship propeller. Presented at the 24th Annual Gallery of Fluid Motion exhibit, held at the 59th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Tampa Bay, November 2006, 2006.

[RLMO03] G. G. Rosa, E. B. Lum, K.-L. Ma, and K. Ono. An interactive volume visualization system for transient flow analysis. In *Proc. Volume Graphics 2003*, pages 137–144, 2003.

[RM00] J. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41(1-2):187–228, 2000.

[RP96] M. Roth and R. Peikert. Flow visualization for turbomachinery design. In *Proc. Visualization 96*, pages 381–384, 1996.

[RP98] M. Roth and R. Peikert. A higher-order method for finding vortex core lines. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 143–150, Los Alamitos, 1998. IEEE Computer Society Press.

[RPS99] F. Reinders, F. Post, and H. Spoelder. Attribute-based feature tracking. In *Data Visualization 1999. Proc. VisSym 99*, pages 63–72, 1999.

[RPS01] F. Reinders, F. Post, and H. Spoelder. Visualization of time-dependent data using feature tracking and event detection. *The Visual Computer*, 17(1):55–71, 2001.

[RSHTE99] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive Exploration of Volume Line Integral Convolution Based on 3D–Texture Mapping. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 233–240, Los Alamitos, 1999.

[RSVP02] F. Reinders, I. A. Sadarjoen, B. Vrolijk, and F. H. Post. Vortex tracking and visualisation in a flow past a tapered cylinder. *Computer Graphics Forum*, 21(4):675–682, Nov. 2002.

[RZNS04] C. Roessl, F. Zeilfelder, G. Nuernberger, and H.-P. Seidel. Reconstruction of volume data with quadratic super splines. *IEEE Trans. Visualization and Computer Graphics*, 10:397–409, 2004.

[SBM$^+$01] G. Scheuermann, T. Bobach, H. H. K. Mahrous, B. Hamann, K. Joy, and W. Kollmann. A tetrahedra-based stream surface algorithm. In *Proc. IEEE Visualization 01*, pages 151 – 158, 2001.

[SBSH04] L. Sun, R. K. Batra, X. Shi, and L. Hesselink. Topology visualization of the optical power flow through a novel c-shaped nano-aperture. In *Proc. IEEE Visualization 2004*, pages 337–344, 2004.

[SGH06] A. Surana, O. Grunberg, and G. Haller. Exact theory of three-dimensional flow separation. Part 1. Steady separation. *Journal of Fluid Mechanics*, 564:57–103, Oct. 2006.

[SH95a] D. Stalling and H. Hege. Fast and resolution independent line integral convolution. *Proceedings Siggraph '95*, pages 249–256, 1995. Los Angeles.

[SH95b] D. Sujudi and R. Haimes. Identification of swirling flow in 3D vector fields. Technical report, Department of Aeronautics and Astronautics, MIT, 1995. AIAA Paper 95-1715.

[SH97] D. Stalling and H.-C. Hege. LIC on surfaces. In *Texture Synthesis with Line Integral Convolution, Siggraph 97, 24. Int. Conf. Computer Graphics and Interactive Techniques*, pages 51–64, 1997. Courses Notes 8.

[SHJK00] G. Scheuermann, B. Hamann, K. Joy, and W. Kollman. Visualizing local vector field topology. *SPIE Journal of Electronic Imaging*, 9(4):356–367, 2000.

[SHK$^+$97] G. Scheuermann, H. Hagen, H. Krüger, M. Menzel, and A. Rockwood. Visualization of higher order singularities in vector fields. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 67–74, 1997.

[Si02] H. Si. Tetgen. a 3D delaunay tetrahedral mesh generator. v.1.2 users manual. WIAS Technical Report No. 4, 2002. http://www.wias-berlin.de/publications/technicalreports/4/.

[SK97] H. Shen and D. Kao. Uflic - a line integral convolution algorithm for visualizing unsteady flows. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 317–323, 1997.

[SKMR98] G. Scheuermann, H. Krüger, M. Menzel, and A. Rockwood. Visualizing non-linear vector field topology. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):109–116, 1998.

[Soi99] P. Soille. *Morphological Image Analysis*. Springer, Berlin, Heidelberg, New York, 1999.

[SP03] M. Sato and R. Peikert. Core-line-based vortex hulls in turbomachinery flows. *J. Visualization Society of Japan*, 23(2):151–154, 2003.

[SP07] F. Sadlo and R. Peikert. Efficient visualization of lagrangian coherent structures by filtered amr ridge extraction. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2007)*, 13(6), November - December 2007.

[SRBE99] M. Schulz, F. Reck, W. Bartelheimer, and T. Ertl. Interactive visualization of fluid dynamics simulations in locally refined cartesian grids. In *Proc. IEEE Visualization 1999*, pages 413–416, 1999.

[SRE05] S. Stegmaier, U. Rist, and T. Ertl. Opening the Can of Worms: An Exploration Tool for Vortical Flows. In *Proc. IEEE Visualization 05*, pages 463–470. IEEE, 2005.

[SS96] D. Stalling and T. Steinke. Visualization of vector fields in quantum chemistry. Technical report, ZIB Preprint SC-96-01, 1996.

[SSZC94] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing features and tracking their evolution. *IEEE Computer*, 27(7):20–27, 1994.

[Sta98] D. Stalling. *Fast Texture-based Algorithms for Vector Field Visualization*. PhD thesis, FU Berlin, Department of Mathematics and Computer Science, 1998.

[STW$^+$06] K. Shi, H. Theisel, T. Weinkauf, H. Hauser, H.-C. Hege, and H.-P. Seidel. Path line oriented topology for periodic 2D time-dependent vector fields. In *Proc. Eurographics / IEEE VGTC Symposium on Visualization (EuroVis '06)*, pages 139–146, Lisbon, Portugal, May 2006.

[SWH05a] J. Sahner, T. Weinkauf, and H.-C. Hege. Galilean invariant extraction and iconic representation of vortex core lines. In K. J. K. Brodlie, D. Duke, editor, *Proc. Eurographics / IEEE VGTC Symposium on Visualization (EuroVis '05)*, pages 151–160, Leeds, UK, June 2005.

[SWH05b] J. Sahner, T. Weinkauf, and H.-C. Hege. Galilean invariant extraction and iconic representation of vortex core lines. In *Proc. EuroVis 2005*, pages 151–160, 2005.

[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):980–990, September - October 2007.

[SX97] D. Silver and X.Wang. Tracking and visualizing turbulent 3D features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, 1997.

[Tak74] F. Takens. Singularities of vector fields. *Publ. Math. IHES*, 43:47–100, 1974.

[TB96] G. Turk and D. Banks. Image-guided streamline placement. In *Proc. Siggraph '96*, pages 453–460, 1996.

[TE99] C. Teitzel and T. Ertl. New approaches for particle tracing on sparse grids. In *Data Visualization 1999. Proc. VisSym 99*, pages 73–84, 1999.

[TF97] H. Theisel and G. Farin. The curvature of characteristic curves on surfaces. *IEEE Computer Graphics and Applications*, 17(6):88–96, 1997.

[TFT05] S. Takahashi, I. Fujishiro, and Y. Takeshima. Interval volume decomposer: a topological approach to volume traversal. In *Visualization and Data Analysis 2005. (Proceedings of the SPIE)*, pages 103–114, 2005.

[TGK$^+$04] X. Tricoche, C. Garth, G. Kindlmann, E. Deines, G. Scheuermann, M. Ruetten, and C. Hansen. Visualization of intricate flow structures for vortex breakdown analysis. In *Proc. IEEE Visualization 2004*, pages 187–194, 2004.

[The95] H. Theisel. *Vector Field Curvature and Applications*. PhD thesis, University of Rostock, November 1995.

[The02] H. Theisel. Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum (Eurographics 2002)*, 21(3):595–604, 2002.

[TIS$^+$95] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. *Computer Graphics Forum*, 14(3):181–192, August 1995.

[TKH00] I. Trotts, D. Kenwright, and R. Haimes. Critical points at infinity: a missing link in vector field topology. In *Proc. NSF/DoE Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization*, 2000.

[TRS03a] H. Theisel, C. Rössl, and H.-P. Seidel. Combining topological simplification and topology preserving compression for 2d vector fields. In *Proc. Pacific Graphics 2003*, pages 419–423, 2003.

[TRS03b] H. Theisel, C. Rössl, and H.-P. Seidel. Compression of 2D vector fields under guaranteed topology preservation. *Computer Graphics Forum (Eurographics 2003)*, 22(3):333–342, 2003.

[TRS03c] H. Theisel, C. Rössl, and H.-P. Seidel. Using feature flow fields for topological comparison of vector fields. In *Proc. Vision, Modeling and Visualization 2003*, pages 521–528, 2003.

[TRW08] H. Theisel, C. Rössl, and T. Weinkauf. Morphological representations of vector fields. In L. Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*, Mathematics and Visualization. Springer, 2008.

[TS03] H. Theisel and H.-P. Seidel. Feature flow fields. In *Data Visualization 2003. Proc. VisSym 03*, pages 141–148, 2003.

[TSH00] X. Tricoche, G. Scheuermann, and H. Hagen. A topology simplification method for 2D vector fields. In *Proc. IEEE Visualization 2000*, pages 359–366, 2000.

[TSH01a] X. Tricoche, G. Scheuermann, and H. Hagen. Continuous topology simplification of planar vector fields. In *Proc. Visualization 01*, pages 159 – 166, 2001.

[TSH01b] X. Tricoche, G. Scheuermann, and H. Hagen. Topology-based visualization of time-dependent 2D vector fields. In *Data Visualization 2001. Proc. VisSym 01*, pages 117–126, 2001.

[TSW$^+$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*, pages 631–638, 2005.

[TTFN05] Y. Takeshima, S. Takahashi, I. Fujishiro, and G. M. Nielson. Introducing topological attributes for objective-based visualization of simulated datasets. In *Eurographics/IEEE VGTC Workshop on Volume Graphics*, pages 137–145, 2005.

[TvW03] A. Telea and J. J. van Wijk. 3D IBFV: Hardware-accelerated 3D flow visualization. In *Proc. IEEE Visualization 2003*, pages 233–240, 2003.

[TW02] H. Theisel and T. Weinkauf. Vector field metrics based on distance measures of first order critical points. In *Journal of WSCG*, volume 10:3, pages 121–128, 2002.

[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*, pages 225–232, 2003.

[TWHS04a] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Grid-independent detection of closed stream lines in 2D vector fields. In *Proc. Vision, Modeling and Visualization 2004*, 2004.

[TWHS04b] 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*, pages 321–328, 2004.

[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):383–394, 2005.

[TWHS07] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. On the applicability of topological methods for complex flow data. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 105–120. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.

[TWSH02] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen. Topology tracking for the visualization of time-dependent two-dimensional flows. *Computers & Graphics*, 26:249–257, 2002.

[VKP99] V. Verma, D. Kao, and A. Pang. Plic: bridging the gap between streamlines and lic. In *Proc. IEEE Visualization 1999*, pages 341–348, 1999.

[VKP00] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Proc. IEEE Visualization 2000*, pages 163–170, 2000.

[vKvOB+97] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. R. Schikore. Contour Trees and Small Seed Sets for Isosurface Transversal. In *Proc. of the 13th ACM Annual Symposium on Computational Geometry (SoCG)*, pages 212–220, 1997.

[vW91] J. van Wijk. Spot noise: Texture synthesis for data visualization. *Computer Graphics*, 25(4):309–318, 1991.

[vW93] J. van Wijk. Implicit stream surfaces. In *Proc. Visualization 93*, pages 245–252, 1993.

[vW02] J. J. van Wijk. Image based flow visualization. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 745–754, 2002.

[vW03] J. J. van Wijk. Image based flow visualization for curved surfaces. In *Proc. IEEE Visualization 2003*, pages 123–130, 2003.

[WEE03] D. Weiskopf, G. Erlebacher, and T. Ertl. A texture-based framework for spacetime-coherent visualization of time-dependent vector fields. In *Proc. IEEE Visualization 2003*, pages 107–114, 2003.

[WGP97] R. Wegenkittl, E. Gröller, and W. Purgathofer. Animating flow fields: Rendering of oriented line integral convolution. In *Proc. Computer Animation '97*, pages 15–21, 1997.

[WGS05] A. Wiebel, C. Garth, and G. Scheuermann. Localized flow analysis of 2d and 3d vector fields. In *Proc. EuroVis 2005*, pages 143–150, 2005.

[WHN+02] T. Weinkauf, H.-C. Hege, B. Noack, M. Schlegel, and A. Dillmann. Flow around a backward-facing step. Winning Entry from the 20th Annual Gallery of Fluid Motion exhibit, held at the 55th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Dallas, November 2002, 2002.

[WHN+03] T. Weinkauf, H.-C. Hege, B. Noack, M. Schlegel, and A. Dillmann. Coherent structures in a transitional flow around a backward-facing step. *Physics of Fluids*, 15(9):S3, September 2003. Winning Entry from the Gallery of Fluid Motion 2003.

[WJE01] R. Westermann, C. Johnson, and T. Ertl. Topology-preserving smoothing of vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):222–229, 2001.

[WLG97] R. Wegenkittl, H. Löffelmann, and E. Gröller. Fast oriented line integral convolution for vector field visualization via the internet. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 309–316, 1997.

[WNC+04] T. Weinkauf, B. Noack, P. Comte, A. Dillmann, and H.-C. Hege. Coherent-structure skeleton of a turbulent mixing layer. Presented at the 22th Annual Gallery of Fluid Motion exhibit, held at the 57th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Seattle, November 2004, 2004.

[WP06] L. Wang and N. Peters. The length-scale distribution function of the distance between extremal points in passive scalar turbulence. *J. Fluid Mechanics*, 554:457–475, 2006.

[WS01] T. Wischgoll and G. Scheuermann. Detection and visualization of closed streamlines in planar flows. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):165–172, 2001.

[WSH01] T. Wischgoll, G. Scheuermann, and H. Hagen. Tracking closed stream lines in time-dependent planar flows. In *Proc. Vision, Modeling and Visualization 2001*, pages 447–454, 2001.

[WST$^+$07] T. Weinkauf, J. Sahner, H. Theisel, H.-C. Hege, and H.-P. Seidel. A unified feature extraction architecture. In R. King, editor, *Active Flow Control*, Notes on Numerical Fluid Mechanics and Multidisciplinary Design (NNFM), pages 119–133. Springer, 2007. Active Flow Control 2006, Berlin, Germany, September 27 - 29.

[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 (Proceedings Visualization 2007)*, 13(6), November - December 2007.

[WT02] T. Weinkauf and H. Theisel. Curvature measures of 3D vector fields and their applications. In *Journal of WSCG*, volume 10:2, pages 507–514, 2002.

[WTHS04a] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Boundary switch connectors for topological visualization of complex 3D vector fields. In *Data Visualization 2004. Proc. VisSym 04*, pages 183–192, 2004.

[WTHS04b] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological construction and visualization of higher order 3D vector fields. *Computer Graphics Forum (Eurographics 2004)*, 23(3):469–478, 2004.

[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):607–616, September 2006. Eurographics 2006, Vienna, Austria, September 04 - 08.

[WTHS07] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Feature flow fields in out-of-core settings. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 51–64. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, September 29 - 30.

[WTS$^+$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*, pages 559–566, 2005.

[YKP05] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3d streamlines. In *Proc. IEEE Visualization 2005*, pages 471–478, 2005.

[ZFN$^+$95] H.-Q. Zhang, U. Fey, B. Noack, M. König, and H. Eckelmann. On the transition of the cylinder wake. *Phys. Fluids*, 7(4):779–795, 1995.

[ZP04] X. Zheng and A. Pang. Topological lines in 3D tensor fields. In *Proc. IEEE Visualization 2004*, pages 313–32, 2004.

[ZPP05] X. Zheng, B. Parlett, and A. Pang. Topological lines in 3d tensor fields and discriminant hessian factorization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):395–407, 2005.

[ZSH96] M. Zöckler, D. Stalling, and H. Hege. Interactive visualization of 3D-vector fields using illuminated stream lines. In *Proc. IEEE Visualization '96*, pages 107–113, 1996.

List of Figures

List of Tables

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Tino Weinkauf
15. Oktober 2007