Computational Discrete Morse Theory

Inauguraldissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften (Dr. rer. nat.)

eingereicht am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

Jan Reininghaus
Zuse-Institut Berlin (ZIB)

Berlin, 2012

Erstgutachterin: Dr. Ingrid Hotz, Zuse Institut Berlin, Germany
Zweitgutachter: Prof. Dr. Thomas Lewiner, PUC Rio, Brazil

Tag der Disputation: 17.02.2012

Abstract

I propose a purely combinatorial framework that allows to extract the extremal structure of scalar and vector fields defined on discrete manifolds. The extremal structure of a scalar field consists of critical points and separatrices – certain tangential curves of the gradient field that connect critical points. The extremal structure of a vector field additionally includes periodic orbits – the tangential curves that are closed. These features are of great interest in many applications since they allow to reduce large data sets to their essential structure.

One of the biggest challenges for classical numerical algorithms is the discrete nature of the extremal structure which necessitates a lot of binary decisions. Their result therefore strongly depends on computational parameters. Since Morse theory relates the extremal structure of a generic function to the topology of the manifold, e.g. by the Poincaré-Hopf Theorem, such numerical methods may thereby compute inadmissible results. Robin Forman has developed a discrete version of Morse theory. This theory can be seen as a discretization of the set of admissible extremal structures for a given manifold.

In my thesis I propose a general computational framework for data analysis which is based on Forman's discrete Morse theory in a graph theoretical formulation. The basic idea is to define a combinatorial optimization problem over the set of admissible extremal structures. The result of this framework is thereby provably consistent with the topology of the domain. Also, the solution of the optimization problem gives rise to a natural hierarchy of extremal structures for a given data set. This hierarchy can be used to remove noise induced extremal structure or to extract its essential extremal structure.

In the context of this unified framework I have developed efficient algorithms and investigated their applicability in 2D for scalar fields [RGH$^+$10], divergence-free vector fields [RH], general vector fields [RH11, RLH11], and time-dependent scalar fields [RKWH11].

Subsequently, this framework has been applied for the analysis of fluid dynamics [KRHH11] and for the computation of a global importance measure for critical points [RKG$^+$11]. It has also been extended to 3D scalar fields [GRP$^+$] and employed for a memory efficient computation of persistent homology [GRWH].

Zusammenfassung

Basierend auf Formans diskreter Morse Theorie schlage ich in meiner Doktorarbeit einen allgemeinen algorithmischen Ansatz zur Datenanalyse in einer graphentheoretischen Formulierung vor. Dieser rein kombinatorische Ansatz erlaubt es, die extremale Struktur von Skalarfeldern und Vektorfeldern, welche auf diskrete Mannigfaltigkeiten definiert sind, zu extrahieren. Die extremale Struktur von einem Skalarfeld besteht aus kritischen Punkten und Separatrizen – bestimmte Tangentialkurven des Gradientenfelds, die die kritischen Punkte verbinden. Die extremale Struktur von Vektorfeldern beinhaltet zusätzlich periodische Orbits – die geschlossenen Tangentialkurven des Vektorfelds. Diese Merkmale sind in vielen Anwendungen von großem Interesse, da sie es ermöglichen, große Datensätze auf ihre essentielle Struktur zu reduzieren.

Eine Herausforderung für klassische numerische Algorithmen stellt die diskrete Natur der extremalen Struktur dar, welche eine große Zahl von Binärentscheidungen bedingt. Das Ergebnis solcher Methoden hängt daher stark von der Wahl der Berechnungsparameter ab. Die Morse Theorie setzt die extremale Struktur einer generischen Funktion zu der Topologie der Mannigfaltigkeit in Beziehung, z.B. im Poincaré-Hopf Theorem. Unter Umständen können klassische numerische Methoden daher unzulässige Ergebnisse erzeugen. Robin Forman hat eine diskrete Version von der Morse Theorie entwickelt. Diese Version kann als eine Diskretisierung der zulässigen extremalen Strukturen einer diskreten Mannigfaltigkeit interpretiert werden.

Die Grundidee meines allgemein algorithmischen Ansatzes zur Datenanalyse ist es, ein kombinatorisches Optimierungsproblem über die Menge der zulässigen extremalen Strukturen zu definieren. Das Ergebnis dieses Ansatzes ist daher immer konsistent in Bezug auf die Topologie von dem Definitionsgebiet. Zusätzlich definiert die Lösung des Optimierungsproblems eine natürliche Hierarchie der extremalen Struktur des Datensatzes. Diese Hierarchie kann benutzt werden, um die von eventuell vorhandenem Rauschen induzierte extremale Struktur zu entfernen, oder, um ausschließlich die essentielle extremale Struktur eines Datensatzes zu extrahieren.

Im Kontext dieses allgemeinen Ansatzes wurden effiziente Algorithmen entwickelt und ihre Anwendbarkeit wurde für zweidimensionale Skalarfelder [RGH+10], divergenzfreie Vektorfelder [RH], allgemeine Vektorfelder [RH11, RLH11] und zeitabhängige Skalarfelder [RKWH11], untersucht.

Die Praxistauglichkeit dieses kombinatorischen Ansatzes zur Datenanalyse wird weiterhin substantiviert durch Anwendungen in der Fluiddynamik [KRHH11] und als Ausgangspunkt für ein globales Wichtigkeitsmaß für kritische Punkte [RKG+11]. Zusätzlich wurde dieser Ansatz auf dreidimensionale Skalarfelder erweitert [GRP+] und diente als Basis für eine Speicherplatz effiziente Berechnung von persistenter Homologie [GRWH].

Danksagung

Zuerst möchte ich meiner Betreuerin Ingrid Hotz danken. Ingrid, ohne deine vorbildliche Führung, deine motivierenden Worte und deine Begeisterungsfähigkeit würde diese Arbeit so nicht existieren. Ich danke dir für viereinhalb schöne Jahre, die mich als Wissenschaftler geprägt haben.

I would also like to thank Thomas Lewiner for an extremely thorough and prompt review of my thesis. Besides being an absolute expert in the field of this thesis, you are a pleasure to talk to. I look forward to many fruitful collaborations with you in the future.

David Günther, Jens Kasten und Christian Löwen möchte ich für die vielen Jahre der Freundschaft und guten Zusammenarbeit danken. Unsere zahlreichen ausgiebigen und anregenden Diskussionen haben die Ergebnisse, die in dieser Arbeit dokumentiert sind, erst ermöglicht.

Sämtlichen Mitarbeitern der Visualisierungsabteilung des Zuse-Instituts Berlin und insbesondere den Mitarbeitern meiner Arbeitsgruppe, Cornelia Auer, Andrea Kratz und Valentin Zobel, sowie Alexander Wiebel möchte ich für eine tolle Arbeitsatmosphäre und die immer vorhandene Hilfsbereitschaft danken.

Ohne meinen Bruder Gerrit hätte ich wohl nie Mathematik studiert. Gerrit, ich danke dir dafür, mich für die Mathematik begeistert zu haben. Auch danke ich dir und Annkathrin für eure Hilfe beim Endspurt.

Meine Eltern Regina und Ralf hatten immer ein offenes Ohr für mich. Ich danke euch für eure unschätzbare Unterstützung.

Glück existiert nur, wenn man es teilt. Mit dir, Nadja, möchte ich mein Glück teilen. Ich danke Dir für deine Geduld, dein Engagement und deine aufmunternden Worten und freue mich auf unsere gemeinsame Zukunft. Ich liebe Dich.

– Jan Reininghaus –

Contents

1 Introduction **1**

2 Related Work **5**
2.1 Vector fields 5
2.2 Scalar fields 6
2.3 Time-dependent scalar fields 7

3 Mathematical Background **9**
3.1 Graph theory 9
3.2 Algebraic topology 18
3.3 Discrete Morse theory 24

4 Computational Discrete Morse Theory **33**
4.1 Graph theoretic formulation 34
4.2 Hierarchical combinatorial vector fields 37
4.3 Generic algorithmic pipeline 39
4.4 Vector fields 43
4.5 Gradient fields 57
4.6 Divergence-free vector fields 63
4.7 Time-dependent scalar fields 69

5 Conclusions and Outlook **87**

Bibliography **91**

List of Figures **101**

List of Tables **103**

Chapter 1

Introduction

We propose a computational framework to extract the extremal structure of scalar and vector fields. The extremal structure of a scalar field consists of critical points and separatrices – the tangential curves of the gradient field that connect critical points whose index differs by one. The extremal structure of a vector field additionally includes periodic orbits – the tangential curves of the vector field that are closed. Examples of extremal structures are shown in Figures 4.7, 4.10 and 4.11. Many more visualizations of extremal structures can found in [Wei08].

These structures are of great interest in many applications and have a long history [Cay59, Max70]. Typically, the critical points are computed by finding all zeros of the (gradient) vector field. The critical points of a scalar field are classified into minima, saddles, and maxima by the eigenvalues of its Hessian, while the critical points of a vector field are classified into sinks, saddles, and sources by the eigenvalues of its Jacobian. The respective eigenvectors can be used to compute the separatrices as the solution of a system of autonomous ordinary differential equations.

This approach to data analysis has to deal with many numerical approximations, e.g. finite floating point precision. Algorithms based on this numerical approach therefore usually contain a number of computational parameters.

One of the biggest challenges that such numerical algorithms face is the discrete nature of the extremal structure which necessitates a lot of binary decisions. For example, the type of a critical point depends on the sign of the eigenvalues. The approximative nature of the analytic approach is thereby further amplified in the context of these binary decisions, where small errors can deal to completely different results. Depending on the input data, the resulting extremal structure may therefore strongly depend on the computational parameters and employed numerical procedures.

From a topological point of view this can be quite problematic. Morse theory [Mil63] relates the extremal structure of a generic function to the topology of its domain, e.g. by the Poincaré-Hopf Theorem, or by the strong Morse

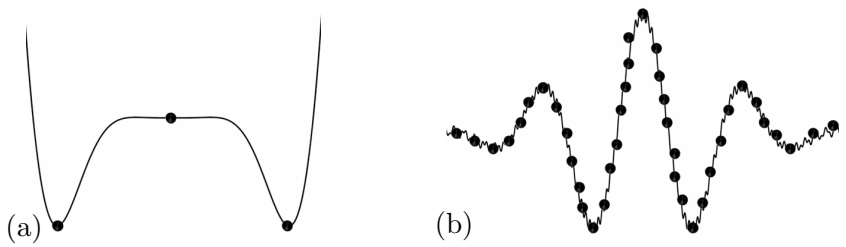


Figure 1.1 Illustration of two algorithmic challenges. (a) shows a 1D function with a plateau-like region. From the topological point of view the critical point in the middle needs to be a maximum since it is located between the two minima on the left and on the right side. However, depending on the numerical procedure the determination of its type might be inconsistent. (b) illustrates a noisy 1D function. Every fluctuation caused by the noise generates additional minima and maxima.

inequalities. For example, a generic scalar field defined on a torus contains at least one minimum, two saddle points, and one maximum. The topology of the domain therefore restricts the set of the admissible extremal structures.

If a (gradient) vector field contains plateau like regions, i.e. regions where the magnitude is rather small, approximative numerical methods may therefore lead to topologically inconsistent results – a small error in an eigenvalue may flip its sign. This means that extremal structures may be computed that cannot exist on the given domain. A simple example for this problem can be given in 1D. Consider an interval containing exactly three critical points as shown in Figure 1.1a. While it is immediately clear that not all critical points can be minima, an algorithm that works strictly locally using numerical algorithms may result in such an inconsistent result.

A second problem that often arises is that the data may contain noise. Depending on its type and quantity, a lot of spurious critical points may be produced as shown in Figure 1.1b. Due to the significance of this problem in practice, a lot of work has been done towards robust methods that can deal with such data, see Chapter 2.

Robin Forman has developed a discrete version of Morse theory [For98a, For98b] for cell complexes. A gradient or vector field is therein directly encoded in the combinatorial structure of the cell complex, and their extremal structure is defined in a combinatorial fashion. A finite cell complex that represents the domain of a function can therefore only carry a finite number of combinatorial (gradient) vector fields, and their respective extremal structure is always consistent with the topology of the domain.

In a certain sense, discrete Morse theory thereby allows to translate the

numerical problem of finding and classifying critical points into a purely combinatorial problem. This approach is thereby a good fit for a computational approach since combinatorial problems can be exactly represented in a computer, whereas numerical approaches have to deal with approximations which are amplified due to binary decisions, see above.

The basic idea of our computational framework is therefore to find a combinatorial (gradient) vector field to represent our input data. Its extremal structure can then be extracted exactly and is by construction always consistent with the topology of the domain – a generic scalar field defined on a torus will always have at least one minimum, two saddle points, and one maximum. This topological consistency greatly improves the robustness of our algorithm. In some sense it serves as an error correcting code: a single misclassification of a critical point cannot occur, as this would result in an inadmissible extremal structure.

For the purpose of data analysis, the computed extremal structure is in general too complex. This is especially true if one deals with noisy data. One is therefore interested in a meaningful and consistent simplification of the extremal structure. Our framework allows for this by computing a hierarchy of combinatorial (gradient) vector fields that represents the input field. The user is then able to select an appropriate level of detail to efficiently analyze the data.

Overall, our combinatorial approach to data analysis has three main advantages compared to the analytic approach:

1. The natural output of this framework is a hierarchy of extremal structures. The importance of a critical point in this hierarchy is determined by a value related to the concept of persistence [ELZ02, EH08]. This allows the user to discriminate between stable and unstable features of the (gradient) vector field.

2. The resulting extremal structure is always consistent with the topology of the underlying domain. While this property may seem rather academic it significantly increases the robustness of the algorithm: a single critical point cannot be missed or misclassified, as this would affect the consistency of the result. In a sense, topological consistency serves as an error correcting code.

3. There are no computational parameters. This enables a fully automatic analysis of a series of (gradient) vector fields and is essential for the treatment of time-dependent data. Furthermore, it significantly simplifies the implementation and verification of the algorithms.

While the proposed framework is applicable to (gradient) vector field data of any dimension, we will often restrict ourselves to 2D manifolds and will only

discuss the applicability and necessary modifications for higher dimensions.

The remainder of this thesis is structured as follows. A small overview of relevant related work in the context of extremal data analysis is given in Chapter 2. The necessary mathematical background in graph theory, algebraic topology, and Morse theory is introduced in Chapter 3. The main ideas of this thesis are presented in Chapter 4. It contains a graph theoretic formulation of discrete Morse theory, a general algorithmic pipeline, describes our hierarchical representation of the extremal structure, and introduces specific algorithms for vector fields, gradient fields, two dimensional divergence-free vector fields, and time-dependent scalar fields.

This thesis is concluded in Chapter 5 with some thoughts on the known shortcomings of our framework, and possible remedies, extensions, and generalizations are discussed.

Chapter 2

Related Work

This chapter gives an overview of existing approaches to data analysis using extremal structures. In contrast to our unified approach, most of these ideas deal only with one specific type of data. We therefore separate the related work into the data types of vector fields, scalar fields, and time-dependent scalar fields.

In general, the use of topological methods for scalar and vector field visualization developed almost independently over the last two decades. Both areas build on solid mathematical foundations. While methods for vector fields mostly refer back to Poincaré Index Theory [Mil65], topological scalar field analysis is based on Morse Theory [Mil63]. For a basic overview over these theories and their relation we refer to the survey [SG08].

2.1 Vector fields

In the context of vector fields, our notion of the extremal structure of a data set is usually referred to as the topology of the vector field. This concept was introduced to the visualization community by Helman and Hesselink [HH89]. They defined the concept of a topological skeleton consisting of critical points and connecting separatrices to segment the field into regions of topologically equivalent streamline behavior. Algorithms to extract periodic orbits, completing this topological structure, have been proposed [WS01, TWHS04, CML+07]. A good introduction to the concepts and algorithms of vector field topology is given in [Wei08].

To improve the applicability of vector field topology, a variety of extensions like topology tracking, extraction of boundary topology, or extensions to 3D have been developed. For a rather complete overview of these methods we refer to the survey paper [LHZP07].

While there are stable numerical algorithms to extract the topological skeleton of a vector field, the overall resulting framework [WST+07] has many computational parameters that may strongly influence the result. To re-

duce the dependence of the algorithms on computational parameters like step sizes, a combinatorial approach to vector field topology based on Conley index theory [Con78] has been developed [CML+07, CMLZ08]. In the case of divergence-free vector fields their algorithm unfortunately encounters many problems in practice.

The topological skeleton is usually rather complex for real world vector fields. One is therefore interested in a meaningful simplification of the skeleton [TSHC01, TSH01, WTS+05, KE07, CML+07].

Since vector field data is often defined in a discrete fashion, a discrete treatment of the differential concepts that are necessary in vector field topology has been shown to be beneficial in [TLHD03, PP02]. They introduced the idea that the critical points of a divergence-free vector field coincide with the extrema of the scalar potential of the point-wise-perpendicular field to the visualization community. The critical points can therefore be extracted by reconstructing this scalar potential and extracting its minima, maxima, and saddle points.

2.2 Scalar fields

In visualization, the extremal structure of a scalar field is often called the topological skeleton. Scalar field topology developed almost independently from vector field topology. The main application areas in visualization include segmentation, transfer function design, and ridge extraction. Due to their robustness and stability, extraction algorithms that assume piece-wise linear data sets have been especially successful in this context [EHNP03, GNP+06, GNPH07].

There are many extraction algorithms [BLW10, KKM05, RWS11, Gyu08, Lew05, Gyu08, GBPH11, GBHP08] that make use of Forman's work [For01] on discrete Morse theory for cell complexes. Rather than choosing a suitable class of continuous functions, a single number is assigned to each cell of the complex and all further steps are combinatorial.

Note that the first implementation of Forman's theory was presented by Lewiner [Lew05, Lew02, LLT04]. His combinatorial gradient vector fields are based on the construction of hypergraphs and hyperforests. In contrast to our computational approach, Lewiner focused on scalar fields, whereas our definitions and algorithms are also directly applicable to vector fields.

To reduce the often very complex topological structure that is generated by the above algorithms, a controlled simplification is frequently introduced based on the mathematically well-founded concept of persistence [ELZ02, EH08]. Due to the simplicity and clarity of this simplification strategy, it has been widely adopted.

2.3 Time-dependent scalar fields

Many algorithms that track features in time-dependent data have been proposed in many different scientific communities. A lot of this work has been partially inspired by object tracking methods in the area of computer vision, see [YJS06] for a survey. In the context of visual data analysis, tracking approaches can roughly be categorized into three classes depending on the treatment of the temporal dimension [Pos03].

The first class considers feature tracking as a two-step process: feature extraction for each time slice and then subsequent feature matching by solving a correspondence. Such methods do not rely on a temporal interpolation. Event analysis mostly happens implicitly during tracking defined by event functions. Common tracked features are volumes or areas, boundaries or contours and points. Correspondence criteria use distance metrics of the domain and the attribute space, which are in general based on application specific heuristics.

Typical attributes comprise feature size, shape descriptors or texture characteristics [CJR07]. Features are linked, if their distance falls below a given threshold [SSZC94, RPS99, LBM$^+$06, dLvL01]. Improvements using feature overlap instead of Euclidean distances are used in [SW97]. A more global approach is followed in [Ji06] employing a best matching algorithm. Improved tracking can be achieved by utilizing additional information for motion prediction [RSVP02]. [BSS02] proposes a progressive tracking of isosurfaces using the isosurface at time t as an initial guess for the next time-step $t + 1$. An extension to tracking the entire contour tree using volume overlap has been proposed in [SB06].

The second class of algorithms considers time as an additional dimension, treated equally to spatial dimensions. Features are extracted from space-time directly, thus increasing the dimension of the domain and the features by one. Tracking is accurate with respect to the chosen temporal interpolation. No explicit distance metrics for features are needed. Event analysis is mostly a subsequent step after tracking and is based on well-founded theory.

Methods extracting iso-surfaces in space-time have been proposed in [WB98, JSW03]. A topological event analysis based on the Reeb-Graph of the surface resulting from sweeping contours has been performed in [WBD$^+$ar, BWP$^+$10]. The development of topological structures in 2D and 3D flow fields has been analyzed in [TWSH02, GTS04]. These algorithms consider vector fields composed of space-time cells with linear interpolation, where events are restricted to cell boundaries. Critical point tracking is thereby reduced to the computation of entry and exit points for each cell. Similarly, [BP02] introduces an algorithm to track vortex core lines over time and scale space searching for features, represented as parallel vectors, on all boundary cells of the space-time cell.

While giving accurate results, these methods are sensitive to noisy data and

a high feature density. To reduce the number of extracted features and events, a common practice is to delete short living features. A combinatorial approach to track critical points is based on the definition of Jacobi sets [EH04]. It consists of Jacobi edges, which are extracted from a spatial-temporal simplicial complex assuming a linear interpolant. The decision whether an edge belongs to the Jacobi curve involves the topological analysis of the lower link of vertices and edges of the simplicial complex. While providing a nice theoretical framework, the resulting Jacobi curves of real data sets are often very complex and hard to analyze. Based on this work it is also possible to track the evolution of the Reeb-graph of a scalar function [EHM$^+$08].

The third class of algorithm combines aspects of both above-mentioned types. They represent the dynamic behavior of features implicitly as streamlines of a higher dimensional derived vector field in space-time. Critical points can then be tracked by computing certain streamlines in this vector field, referred to as a Feature Flow Field [TS03]. Recently, a combinatorial version of the Feature Flow Field method has been proposed [KKM08]. This method is discussed in detail in Section 4.7.1 and provides the mathematical foundation for our treatment of time-dependent data presented in Section 4.7.2.

Chapter 3

Mathematical Background

The goal of this chapter is to introduce the relevant mathematical background which will be made use of in Chapter 4. We will present some basic ideas and results from graph theory, algebraic topology, and discrete Morse theory in a way that assumes very little prior knowledge in these areas.

Note that the presented material does not include any justifications, proofs, or detailed complexity and correctness analysis of the presented algorithms. We will instead refer the interested reader to relevant books on these topics that give a complete and formally correct description of this material. Also, we will generally only present the relevant special case of a general result to keep the presentation both brief and focused.

3.1 Graph theory

In this section we introduce some basic notions from graph theory. For a more general introduction to graph theory that contains proofs and justifications we refer to [Die97, Sch03].

The required formalism for the algorithms presented in Chapter 4 that analyze the extremal structure of discrete data sets can be motivated with a simple example:

Suppose we are given a set of workers and a set of projects. Further suppose that each worker is able to process some of the projects with a varying degree of efficiency. Each project can be processed by at most one worker, and each worker can only work on one project. The goal is now to find an assignment of workers to projects that maximizes the overall efficiency.

In the following sections, we will briefly present a mathematical model for this kind of problem and introduce an algorithm that can be used to compute the optimal assignment of workers to projects efficiently.

3.1.1 Basic definitions

In this section some basic notions and algorithms from graph theory is introduced. We focus on the material that is necessary for the ideas presented in Chapter 4.

Loosely speaking, a graph is a set of nodes that are connected by a set of edges. To be more precise, let N denote a finite set and $E \subseteq \{L \subseteq N : |L| = 2\}$, where $|\cdot|$ denotes the size of a set. Then the tuple $G = (N, E)$ is called a ***simple graph***. Since we will mostly deal with simple graphs, we will refer to them just as graphs. We will denote the number of nodes $|N|$ by n, and the number of edges $|E|$ by m. If the nodes of the edges are ordered, i.e. $\vec{E} \subseteq V \times V$, then $D = (N, \vec{E})$ is called a ***directed graph***. Examples of (directed) graphs are shown in Figure 3.1.

Figure 3.1 Graphs. (a) a simple graph $G = (N, E)$ with $N = \{a, b, c, d\}$ and $E = \{\{a, b\}, \{c, a\}, \{c, b\}, \{c, d\}\}$. (b) a directed graph $D = (N, \vec{E})$ with $N = \{a, b, c, d\}$ and $\vec{E} = \{(a, b), (a, c), (b, c), (c, d)\}$

Throughout this thesis, we will often consider certain subsets of a given graph $G = (N, E)$. We therefore introduce two useful notational shortcuts. Let $\mathcal{P}(\cdot)$ denote the power set. If $U \subseteq N$ is a subset of nodes, then $E(U) = E \cap \mathcal{P}(U)$ denotes the subset of edges whose nodes are contained in U. If $L \subseteq E$ denotes a subset of edges, then $N(L) = \bigcup_{e \in L} e$ denotes the subset of nodes that are part of at least one edge in L.

A ***path*** p in a graph is a sequence of unique nodes $u_0 u_1 \dots u_\ell$ such that $\{u_k, u_{k+1}\} \in E$ for $k = 0, \dots, \ell - 1$. A sequence of such nodes $u_0 u_1 \dots u_\ell$ where $u_1 \dots u_{\ell-1}$ are unique and $u_0 = u_\ell$ is called a ***cycle***. Note that a path (cycle) can as well be represented by a sequence of edges. We will always implicitly make use of the applicable representation. Examples of paths and cycles are shown in Figure 3.2.

The length of a path $u_0 u_1 \dots u_\ell$ is defined as the number of participating edges: $\ell - 1$. A graph is called ***connected*** if there is a path connecting each pair of nodes. Note that most algorithms in graph theory assume without loss of generality that the graph is connected, since they could as well be applied to each connected component of the graph individually. In this thesis we will always implicitly assume that the given graph is connected.

In most algorithms that deal with graphs we need to traverse the graph – visiting each node once. A simple algorithm that is often used is the ***breadth***

Figure 3.2 Paths and cycles. (a) a path *abcd*. (b) a cycle *abca*.

first search. The pseudo code of this algorithm is shown in Algorithm 1 using a queue data structure Q with standard push and pop operations. The input of this algorithm consists of a simple graph $G = (N, E)$ and a start node $s \in N$. It can be implemented with a running time of $O(n + m)$. An illustration of the breadth first search algorithm is shown in Figure 3.3.

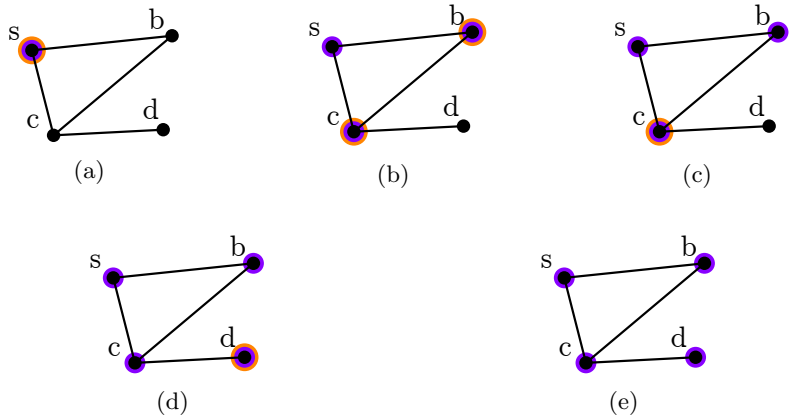


Figure 3.3 Breadth first search algorithm. (a)-(e) illustration of Algorithm 1 – the blue disks represents the content of the set U, while orange shows the contents of the queue Q.

A simple application of this algorithm is to test a graph for connectedness. We simply start Algorithm 1 with an arbitrary node of the graph. The graph is then connected iff $U = N$ at the end of the algorithm.

Note that if one replaces the queue data structure with a stack in Algorithm 1, then the resulting algorithm is called ***depth first search***.

3.1.2 Shortest paths

A common application of graph theory is the computation of distances in some network modeled as a directed graph $D = (N, \vec{E})$ with edge weights $\omega : \vec{E} \to \mathbb{R}$ to represent the cost of traversing an edge. In such a directed ***edge weighted graph*** $D = (N, \vec{E}, \omega)$, the weight of a path (or cycle) is defined as the sum of the weights of the participating edges. The distance of two nodes in

Algorithm 1 Breadth first search

Input: $G = (N, E), s \in N$

```
1: U ← s
2: Q ← s
3: while Q ≠ ∅ do
4:    u ← pop(Q)
5:    for all {{u, w} ∈ E : w ∉ U} do
6:       U ← U ∪ w
7:       push(Q, w)
```

Algorithm 2 Bellman Ford

Input: $D = (N, \vec{E}, \omega), s \in N$
Output: $\text{dist} : N \to \mathbb{R}$, $\text{pred} : N \to N$

```
1: for all v ∈ N do
2:    dist(v) ← ∞
3:    pred(v) ← v
4: dist(s) ← 0
5: for k = 1 to |N| − 1 do
6:    for all (u, w) = e ∈ E⃗ do
7:       if dist(u) + ω(e) < dist(w) then
8:          dist(w) ← dist(u) + ω(e)
9:          pred(w) ← u
```

a weighted graph is then defined as the minimal weight the paths connecting them.

If the weighted graph does not contain any cycle with negative weight one can use the Bellman-Ford algorithm [Bel58, For56] to compute the distances from one node to every other node as shown in Algorithm 2. The input of this algorithm consists of a weighted directed graph $D = (N, \vec{E}, \omega)$ and a start node $s \in N$. The output consists of the distance function $\text{dist} : N \to \mathbb{R}$ and a predecessor function $\text{pred} : N \to N$. $\text{pred}(\cdot)$ can be used to construct a minimal path from every node v to the start node s by a repeated application of $\text{pred}(\cdot)$ to v. A straightforward analysis reveals that Algorithm 2 has a running time of $O(nm)$. An illustration of the Bellman-Ford algorithm is shown in Figure 3.4.

3.1.3 Matchings

The most central notion that is used in Chapter 4 will now be introduced: a pairwise disjoint subset of edges $M \subseteq E$ of a graph $G = (N, E)$ is called a ***matching***. The set of all matchings is denoted by $\mathcal{M} = \{L \subseteq E : \ell_1 \cap \ell_2 = \emptyset, \text{ for all } \ell_1 \neq \ell_2 \in L\}$. An example of a matching is shown in Figure 3.5.

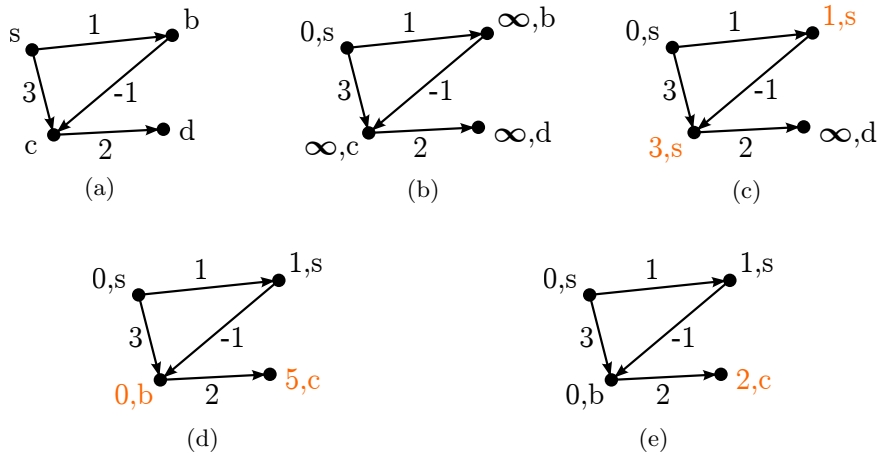


Figure 3.4 Bellman-Ford algorithm. (a) a directed edge weighted graph. (b)-(e) illustration of Algorithm 2 – the values of the functions dist(·) and pred(·) are shown adjacent to each node and updates are highlighted in orange.

Given an edge weight function ω, the weight of a matching M is defined as the sum of weights of the edges contained in M. Given an edge weighted graph $G = (N, E, \omega)$ we can then define the following optimization problem

$$L = \arg\max_{M \in \mathcal{M}} \omega(M), \tag{3.1}$$

i.e., we want to find the matching in G which maximizes the weight function. This optimization problem is usually referred to as the ***maximum weighted matching*** problem. If there exists a disjoint decomposition $N = U \dot{\cup} W$ of the nodes of a graph G such that $E(U) = E(W) = \emptyset$, then G is called ***bipartite***. An example of a bipartite graph is shown in Figure 3.5. For graphs with such a structure (3.1) is called the maximum weighted bipartite matching problem.

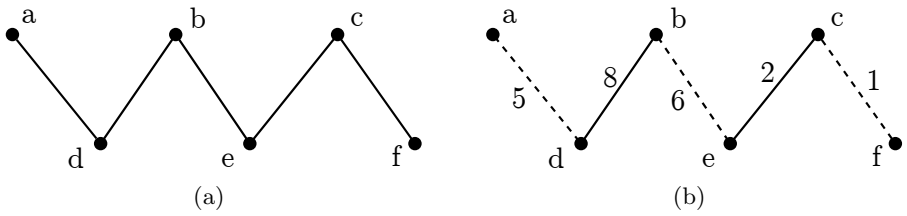


Figure 3.5 Bipartite matchings. (a) the shown graph $G = (N, E)$ is bipartite since $N = U \dot{\cup} W$ with $U = \{a, b, c\}$, $W = \{d, e, f\}$ and $E(U) = E(W) = \emptyset$. (b) the set of dashed edges $M \subseteq E$ defines a matching in G since no two dashed edges meet at a node. Its weight $\omega(M)$ is $5 + 6 + 1 = 12$

The formalism that has been presented so far can be used to formalize the model optimization problem introduced in the beginning of Section 3.1:

The set of workers and projects are represented as nodes in a graph. The ability of the workers to process some of the projects with a varying degree of efficiency can be modeled by weighted edges in this graph. Note that there are no edges connecting a worker (project) to another worker (project). The resulting graph is therefore bipartite. The restriction that each project can be worked on by at most one worker, and each worker can only work on one project implies that the valid assignments are given by the set of matchings in this graph. To compute an optimal assignment we therefore need to find the solution of (3.1).

We will now outline an algorithm that solves such problems. The basic idea of this method is to compute the sequence

$$L_k = \arg \max_{M \in \mathcal{M},\, |M|=k} \omega(M). \tag{3.2}$$

The maximum weight matching is then given by $L = \arg\max \omega(L_k)$. Since $L_0 = \emptyset$, it suffices to develop an algorithm that can compute L_{k+1} given L_k. To describe this algorithm we first introduce some useful concepts.

An ***alternating path*** in a graph G is defined as a path whose edges alternate with respect to a matching M. If the start and end node of an alternating path are unmatched, i.e. they are not contained in $N(M)$, then it is called an ***augmenting path***. The augmenting paths are called 'augmenting' because they can be used to increase the size of a matching: if p is an augmenting path with respect to a matching M with k edges, then $M \,\Delta\, p$ is a matching with $k+1$ edges. Here and throughout this thesis, Δ denotes the symmetric difference operator $A \,\Delta\, B = (A \cup B) \setminus (A \cap B)$. An illustration of an augmenting path is shown in Figure 3.6.

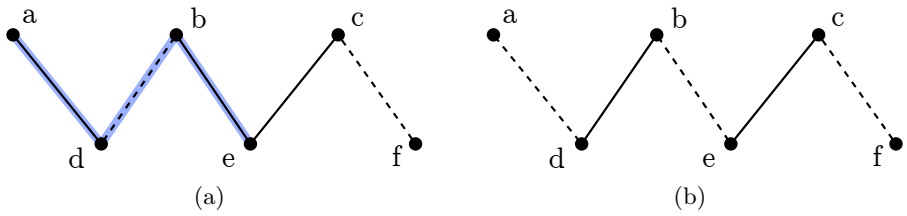


Figure 3.6 Augmenting path. (a) a bipartite graph $G = (N, E)$ with a matching $M \subseteq E$ (dashed lines). The path p indicated by the blue highlight is an augmenting path for this matching since its edges alternate between M and $E \setminus M$ and its start- and endpoint are not covered by M. (b) the augmented matching $\tilde{M} = M \,\Delta\, p$.

The weight of an augmenting path $p = e_0 e_1 \dots e_k$ is defined as the alternating sum of its edge weights, i.e. $\omega(p) = \sum_{\ell=0}^{k} (-1)^\ell \omega(e_\ell)$. Note that using this convention we have $\omega(M \,\Delta\, p) = \omega(M) + \omega(p)$.

Algorithm 3 Matching induced directed graph

Input: $G = (U \dot\cup W, E, \omega), M \in \mathcal{M}$
Output: $D = (U \dot\cup W \dot\cup \{s,t\}, \vec{E}, \vec{\omega})$

for all $e = \{u,w\} \in E \setminus M, u \in U, w \in W$ **do**
  $\vec{E} \leftarrow \vec{E} \cup (u,w)$
  $\vec{\omega}((u,w)) \leftarrow -\omega(e)$
for all $e = \{u,w\} \in M, u \in U, w \in W$ **do**
  $\vec{E} \leftarrow \vec{E} \cup (w,u)$
  $\vec{\omega}((w,u)) \leftarrow \omega(e)$
for all $u \in U \setminus U(M)$ **do**
  $\vec{E} \leftarrow \vec{E} \cup (s,u)$
  $\vec{\omega}((s,u)) \leftarrow 0$
for all $w \in W \setminus W(M)$ **do**
  $\vec{E} \leftarrow \vec{E} \cup (w,t)$
  $\vec{\omega}((w,t)) \leftarrow 0$

The ***Hungarian method*** [Kuh55] now makes use of the fact that we can increase the size of a maximum weighted matching by augmenting it with the heaviest augmenting path, i.e.

$$L_{k+1} = L_k \,\Delta \arg\max_{p \in \mathcal{A}_{L_k}} \omega(p), \tag{3.3}$$

where $\mathcal{A}_M$ denotes the set of all augmenting paths in a graph with respect to a given matching M. We refer to [Sch03] for a proof. Essentially, the Hungarian method exploits the fact that – in the context of augmenting paths – the optimization problem (3.2) can be solved in a greedy fashion.

The main task is therefore the computation of $\arg\max_{p \in \mathcal{A}_M} \omega(p)$ for a given matching M. Such a path can be computed by solving a shortest path problem which we now describe. We first define a directed graph which is induced by M and the bipartition $N = U \dot\cup W$ using Algorithm 3. We first orient all edges between U and W according to the matching M and assign an appropriate weight to them. Then, we insert two auxiliary nodes s, t and edges with weight 0 that connect them to U and W respectively. An example of this construction can be seen in Figure 3.7a and Figure 3.7b.

We can now apply the shortest path Algorithm 2 to the directed graph D with the start node s. An augmenting path of maximum weight can now be computed by a repeated application of the resulting $\text{pred}(\cdot)$ function to the node t – we only have to remove the auxiliary start and end nodes s and t from the resulting path. If $\text{pred}(t) = t$, then there is no augmenting path, so the matching has maximum cardinality and the algorithm is finished since it has computed the whole sequence (3.2). An illustration of a single iteration of the Hungarian method is shown in Figure 3.7.

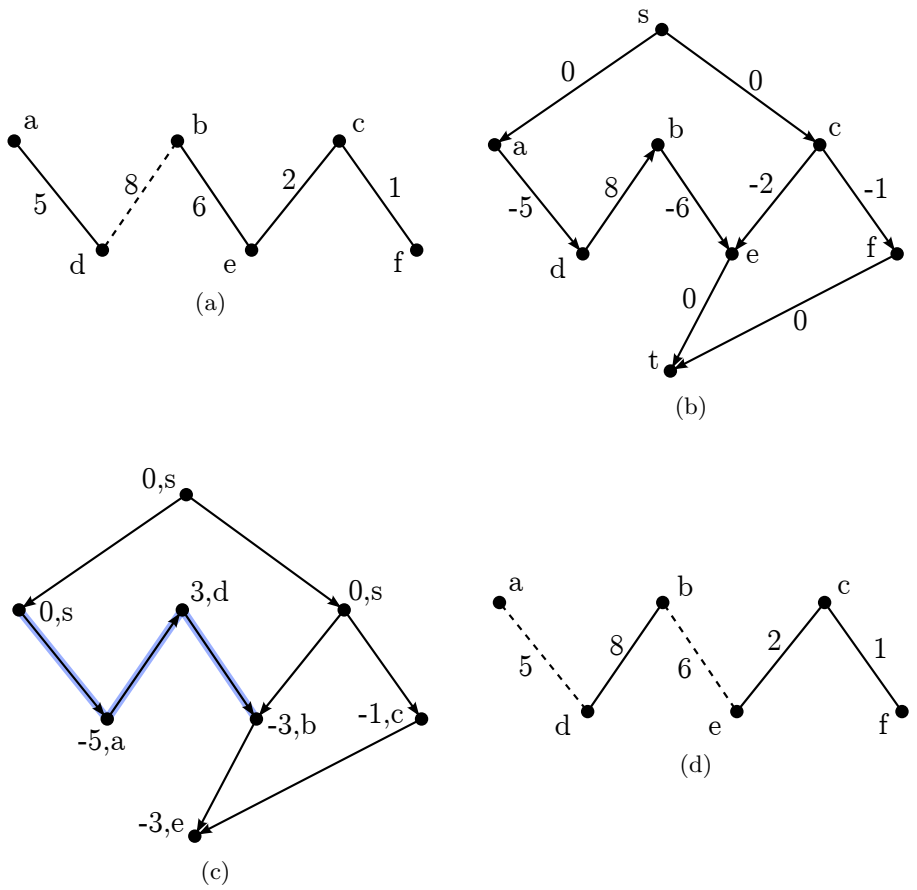


Figure 3.7 Hungarian method. (a)-(d) a single iteration of the Hungarian method. (a) input: the maximum weight matching containing one edge. (b) the induced directed graph of the matching of (a) computed with Algorithm 3. (c) the result of applying the Bellman-Ford algorithm to (b). The interior p of the shortest path from s to t is shown in blue – it is the heaviest augmenting path in (a). (d) output: the maximum weight matching with two edges is given by the symmetric difference of the matching in (a) with the augmenting path computed in (c).

Note that the Bellman-Ford algorithm only works when there is no cycle with negative weight. Fortunately, the induced directed graphs generated by Algorithm 3 do not contain such cycles due to the maximality property of the matchings (3.2): suppose that the directed graph induced by the matching L_k would contain a cycle p with negative weight. Considered as an augmenting path, p has therefore positive weight. Then $L_k \Delta p$ is again a matching with k edges, and $\omega(L_k) < \omega(L_k \Delta p)$ in contradiction to the maximality property of L_k defined in (3.2).

The size of a matching and the length of augmenting paths is bounded by n, since a node can only be part of one matching edge. The running time of the Hungarian method – using the Bellman-Ford algorithm – is therefore given by $O(n^2m)$.

Note that it is possible to use Dijkstra's algorithm [Dij59] to compute the augmenting path of maximum weight even though the directed graph contains edges with negative weight. The basic idea is to shift the edge weights with a potential and maintain it during the augmentation phase. This results in a total running time of $O(n^2 \log n + nm)$ using a Fibonacci heap [FT87]. For more details on this approach we refer to [Sch03].

3.2 Algebraic topology

In this section we will introduce some basic notions from algebraic topology that will be made use of in Chapter 4. For a more general and formally correct introduction to algebraic topology that contains proofs and justifications we refer to [Hat02, Mun84].

The purpose of the presented material is twofold. First, it gives a formal description of the type of input that is supported in computational discrete Morse theory. Secondly, it is essential to appreciate the consistency that the algorithmic approach presented in Chapter 4 offers.

Loosely speaking, topology is concerned with the structure of a shape that does not depend on its exact form. For example, a donut and a coffee cup are considered to have the same structure – they can be deformed into each other without cutting or gluing. In contrast, a torus and a sphere are considered to be fundamentally different, see Figure 3.8 for an example. In the next section we give a brief but concise formal description of this idea.

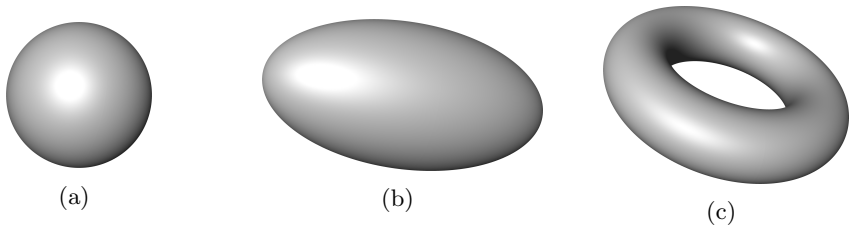


Figure 3.8 Surfaces. (a) a sphere. (b) an ellipsoid. (c) a torus. (a) and (b) are considered to be topologically equivalent since they can be deformed into each other. In contrast, (c) has a fundamentally different structure compared to (a) or (b).

3.2.1 Basic definitions

Let $\mathcal{P}(\cdot)$ denote the power set. A ***topological space*** is a set X together with a set of its subsets $\mathcal{X} \subseteq \mathcal{P}(X)$, called open sets, that satisfies

1. $\emptyset \in \mathcal{X}$ and $X \in \mathcal{X}$,
2. $\bigcup_{A \in \mathcal{A}} A \in \mathcal{X}$ for any $\mathcal{A} \subseteq \mathcal{X}$,
3. $\bigcap_{A \in \mathcal{A}} A \in \mathcal{X}$ for any $\mathcal{A} \subseteq \mathcal{X}$ with $|\mathcal{A}| < \infty$.

A common example of a topological space is $\mathbb{R}$. The open sets of $\mathbb{R}$ are often defined as the sets which are unions of open intervals (a, b).

Let $(X, \mathcal{X})$ and $(Y, \mathcal{Y})$ denote topological spaces. A function $f : X \to Y$ is called ***continuous*** if the inverse image of any open set is an open set, i.e.

$f^{-1}(A) \in \mathcal{X}$ for any $A \in \mathcal{Y}$. Note that when $X = Y = \mathbb{R}$, this definition of a continuous function coincides with the usual $\epsilon - \delta$ condition encountered in basic calculus classes.

If a continuous function f is one-to-one, onto, and its inverse function f^{-1} is also continuous, then f is called a ***homeomorphism***. Two topological spaces X and Y are called homeomorphic iff there exists a homeomorphism $f : X \to Y$.

Let us consider two simple examples to illustrate these concepts:

1. The set $X = (-2, 2) \subseteq \mathbb{R}$ is homeomorphic to the set $Y = (-8, 8)$ via the homeomorphism $x \mapsto x^3$.
2. The set $X = (-1, 1)$ is not homeomorphic to $Y = (-2, -1) \cup (1, 2)$ since any continuous function maps connected sets to connected sets. Since X is connected and Y is not, there cannot exist a homeomorphism $f : X \to Y$.

One of the central questions in topology is whether two given topological spaces X and Y are homeomorphic. Unfortunately, this is almost never as easy as in the preceding two examples – even proving that $\mathbb{R}^n$ is not homeomorphic to $\mathbb{R}^{n+1}$ is far from trivial.

A common approach to show that two topological spaces are not homeomorphic is to make use of topological invariants. A topological invariant is a property that is preserved under homeomorphisms – if it associates different values to X and Y then those spaces cannot be homeomorphic. A simple example of a topological invariant is the number of connected components. In the following section we will introduce a relatively simple, yet powerful topological invariant.

3.2.2 Simplicial homology with $\mathbb{Z}_2$ coefficients

In this chapter we restrict ourselves to very simple objects to keep the presented concepts as simple as possible. These objects, called simplicial complexes, arise in many applications like numerical simulations or computer graphics. While the material introduced below is in principal applicable to higher dimensions, we will only make use of two dimensional examples since this simplifies notation drastically.

Note that our presentation of simplicial homology using $\mathbb{Z}_2$ coefficients is a lot less powerful than the standard introduction to homology with coefficients in $\mathbb{Z}$. The main advantage of our approach lies instead in its simplicity – most algebraic concepts that usually need to be introduced for standard homology are not necessary in this degenerate case of homology.

The basic building blocks we want to consider to represent some geometric object are simplices. Let $\{v_0, \ldots, v_k\}$ denote a set of points in general position,

i.e. the set of vectors $\{v_1 - v_0, \ldots, v_k - v_0\}$ is linear independent. The convex hull of $\{v_0, \ldots, v_k\}$ is called a ***k-simplex***. Figure 3.9 depicts examples of k-simplices. We will denote the set of all k-simplices by S_k. For example, S_0 consists of all points, S_1 of all edges, and S_2 of all triangles.

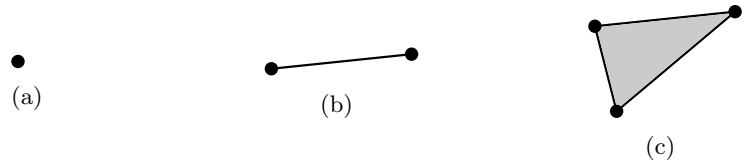


Figure 3.9 Simplices. (a) a 0-simplex. (b) a 1-simplex. (c) a 2-simplex.

Note that a k-simplex α is always the convex hull of $k+1$ points. The convex hull of any subset of these points is called a ***face*** of α. We denote the set of all faces of a simplex α by $F(\alpha)$. For example, if α is a triangle, then $F(\alpha)$ consists of three edges, three points, and the empty set.

A finite set of simplices $K = \bigcup_{k=0}^{d} K_k$, $K_k \subseteq S_k$, is now called a d-dimensional finite ***simplicial complex*** if

1. $F(\alpha) \subseteq K$, for any $\alpha \in K$,
2. $\alpha \cap \beta \in F(\alpha) \cap F(\beta)$, for any $\alpha, \beta \in K$.

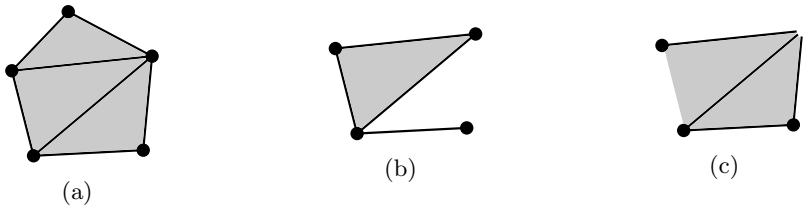
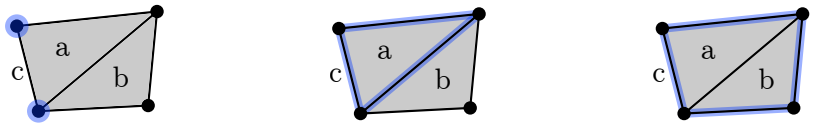


Figure 3.10 Simplicial complex. (a)-(c) finite sets of simplices. (a) and (b) are simplicial complexes. (c) is not a simplicial complex since not all faces of each simplex are present.

The definition of a simplicial complex is illustrated in Figure 3.10. Common examples of simplicial complexes that are often encountered in practice are graphs, as introduced in Section 3.1, triangulations of surfaces, or tetrahedizations of volumetric shapes.

Let C_k denote the power set of K_k. We will now define a ***boundary operator*** ∂_k that maps an element of C_k to C_{k-1}. The boundary of a single k-simplex α is defined as the $(k-1)$ dimensional faces, i.e. $\partial_k \alpha = F(\alpha) \cap K_{k-1}$. The boundary of a set of k-simplices is now defined using the symmetric difference operator: $\partial_k \{\alpha_0, \alpha_1, \ldots, \alpha_\ell\} = \partial_k \alpha_0 \,\Delta\, \partial_k \alpha_1 \,\Delta \ldots \Delta\, \partial_k \alpha_\ell$. Figure 3.11 illustrates this definition of a boundary operator.



(a) (b) (c)

Figure 3.11 Boundary operator. (a) the boundary of c consists of the 0-simplices highlighted in blue. (b) the boundary of the 2-simplex a consists of the 1-simplices highlighted in blue. (c) the boundary of $\{a, b\}$ consists of the 1-simplices highlighted in blue – the interior 1-simplex drops out due to the symmetric difference operator.

Since direct calculation shows that for any k-simplex α we have $\partial_{k-1}\partial_k\alpha = \emptyset$ the sequence

$$C: \quad \emptyset \xrightarrow{\partial_{d+1}} C_d \xrightarrow{\partial_d} C_{d-1} \xrightarrow{\partial_{d-1}} \ldots \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} \emptyset \tag{3.4}$$

is called a ***chain complex***. The auxiliary boundary maps ∂_{d+1} and ∂_0 are defined by $\partial_{d+1}\emptyset = \emptyset$ and $\partial_0\alpha = \emptyset$ for any simplex $\alpha \in C_0$.

Given such a chain complex we can define its homology. Let $Z_k = \ker \partial_k = \{\alpha \in C_k : \partial_k\alpha = \emptyset\}$ denote the set of k-cycles, and $B_k = \operatorname{im} \partial_{k+1} = \{\alpha \in C_k :$ there exists β with $\partial_{k+1}\beta = \alpha\}$ the set of k-boundaries. We now define an equivalence relation on Z_k: $x \sim y$ iff $x \,\Delta\, y \in B_k$. The number of non-trivial equivalence classes in $H_k = Z_k/\sim$ is called the k-th ***Betti number*** b_k of K. When we think of the symmetric difference operator Δ as a group operation, we can call H_k the kth ***homology group*** of K. For an illustration of the concept of homology we refer to Figure 3.12.

It can be shown that the homology groups (and subsequently the Betti numbers) are topological invariants [Hat02]. This implies that if the homology groups of two simplicial complexes differ, then the respective topological spaces cannot be homeomorphic.

For example, let K denote a simplicial complex representing a torus and let L denote a simplicial complex representing a sphere. Using a small number of simplices one can manually calculate the Betti numbers of K: $(b_0 = 1, b_1 = 2, b_2 = 1)$, and of L: $(b_0 = 1, b_1 = 0, b_2 = 1)$. As the Betti numbers do not coincide we know that a sphere and a torus are not homeomorphic.

Since Betti numbers are topological invariants, so is their alternating sum

$$\chi(K) = b_0 - b_1 + \ldots + (-1)^d b_d. \tag{3.5}$$

$\chi(K)$ is called ***Euler characteristic*** of K. Using basic tools from homological algebra [Mun84] one can show that

$$b_0 - b_1 + \ldots + (-1)^d b_d = |K_0| - |K_1| + \ldots + (-1)^d |K_d|. \tag{3.6}$$

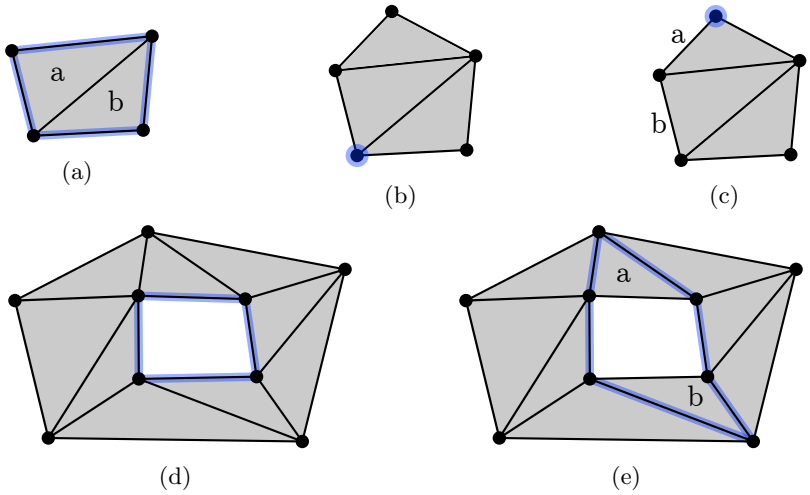


Figure 3.12 Homology. (a) a 1-cycle $x \in Z_1$. It is equivalent to the empty set since $x \,\Delta\, \emptyset = x = \partial\{a, b\} \in B_1$. (b) a 0-cycle. (c) another 0-cycle x. It is equivalent to the cycle y shown in (b) since $x \,\Delta\, y = \partial\{a, b\} \in B_0$. (d) a 1-cycle x. It is not equivalent to the empty set since $x \,\Delta\, \emptyset = x \notin B_1$. (e) another 1-cycle y. It is equivalent to the cycle x shown in (d) since $x \,\Delta\, y = \partial\{a, b\} \in B_1$.

The Euler characteristic is therefore very easy to compute – one just has to count the different types of simplices in a simplicial complex. Perhaps surprisingly, the Euler characteristic is still a very descriptive topological invariant: two closed orientable surfaces are homeomorphic iff they have the same Euler characteristic.

Note that the material presented in this section is in principal applicable to more general representations of topological spaces than simplicial complexes. One of the most widely used generalizations is called the ***regular cell complex***. Since a formally correct definition of a regular cell complex is quite involved we refer the interested reader to standard literature [Hat02] in algebraic topology for a precise definition.

For our purposes, it suffices to say that regular cell complexes are structurally similar to simplicial complexes but allow for more general geometric primitives. Instead of building the complex from k-simplices one makes use of k-cells, which are primitives homeomorphic to k-balls. A simple example of a k-cell is thereby a k-simplex. A more complex example of a k-cell is a k-polytope.

The notion of a cell complex is thereby very general and covers triangulations, quadrangulations, and meshes that consist of prisms. This generalization of a simplicial complex is therefore also quite useful in practice since many meshes are cell complexes but not simplicial complexes. Even if the original mesh is a simplicial complex we may encounter a cell complex as an

auxiliary construct, as in Section 3.3.3 or Section 4.7.

Since the general idea of homology is however already well explained using simplices, we chose this simple approach to keep the technical overhead at a minimum. The ideas presented in Section 3.3.2 will follow a similar approach – we only present the definitions and results for simplicial complexes even though the results are also valid for regular cell complexes.

3.3 Discrete Morse theory

This section introduces the basic ideas from Forman's discrete Morse theory [For98b, For98a]. The presented definitions and ideas will be essential for the main chapter of this thesis in which we will develop a computational approach to discrete Morse theory and apply it to problems in data analysis.

We begin with a short and informal introduction to smooth Morse theory [Mil63] to motivate the applicability of the discrete approach.

3.3.1 Morse theory

The basic idea in Morse theory [Mil63] is to investigate the relationship of the critical points of a smooth function $f : \Omega \to \mathbb{R}$ with the topology of Ω. A point $x \in \Omega$ is called a ***critical point*** of index k iff the gradient of f is zero at x and the Hessian of f at x has k negative eigenvalues. In two dimensions, a critical point of index 0 is called a minimum, of index 1 a saddle, and of index 2 a maximum.

If the determinant of the Hessian at a critical point x is nonzero, then x is called a non-degenerate critical point. If all critical points of f are non-degenerate, then f is called a ***Morse function***.

Using these concepts we can now state a special case of one of the main theorems in Morse theory: if the homology groups of the sub-level sets $f^{-1}((-\infty, a])$ and $f^{-1}((-\infty, b])$ of a Morse function f differ, then $f^{-1}([a, b])$ contains at least one critical point. This theorem is illustrated for a simple function in Figure 3.13.

This theorem gives a perhaps surprising relation between topology and analysis. In fact, the theorem implies that in a discrete setting one can even define critical points in a purely topological fashion. And since homology groups can be computed in an algebraic setting this relation is quite useful for computational purposes.

Not only is Morse theory useful for this alternative definition of a critical point, it also characterizes the admissible set of critical points of a function defined on Ω. For example, every Morse function defined on a torus must have at least four critical points. We will give a precise definition of such topological constraints on the set critical points in Section 3.3.2 in the discrete setting.

The most fine grained topological invariant we want to discuss, called Morse homology, requires the notion of the Morse-Smale complex [Sma61] which is only defined for a certain set of non-degenerate functions. A path $p : [0, 1] \to \Omega$ is called an integral line of f if it is tangential to the gradient field ∇f. An integral line connecting two critical points of a Morse function whose index differs by one is called a ***separatrix***. If there are no integral lines connecting critical points of the same index, then f is called a ***Morse-Smale function***. Note that if we slightly perturb a non-Morse-Smale function, then the result

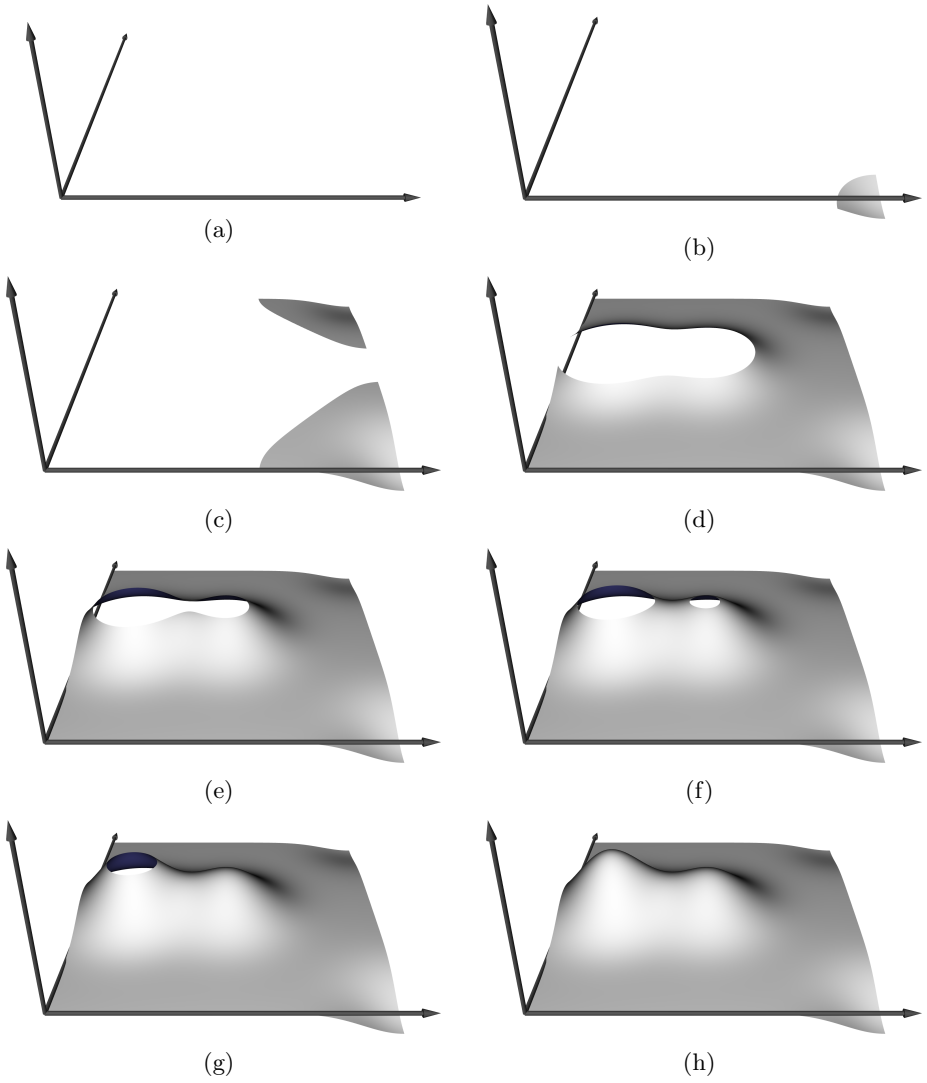


Figure 3.13 Morse theory. (a)-(h) the sub-level set $F^y = f^{-1}((-\infty, y])$ of a smooth function f for different values of y visualized using height fields. (a) F^y is the empty set. (b) a component is created. (c) another component is created. (d) the two components have merged. (e) a hole is created. (f) another hole is created. (g) a hole is destroyed. (h) the second hole is destroyed.

will be a Morse-Smale function. In a certain sense, almost all functions are thereby Morse-Smale.

The Morse-Smale complex associated to a Morse-Smale function can now be constructed using the critical points and separatrices of f. The details of this construction are described in the following sections for the discrete case. A simple example of Morse-Smale complex is depicted in Figure 3.14.

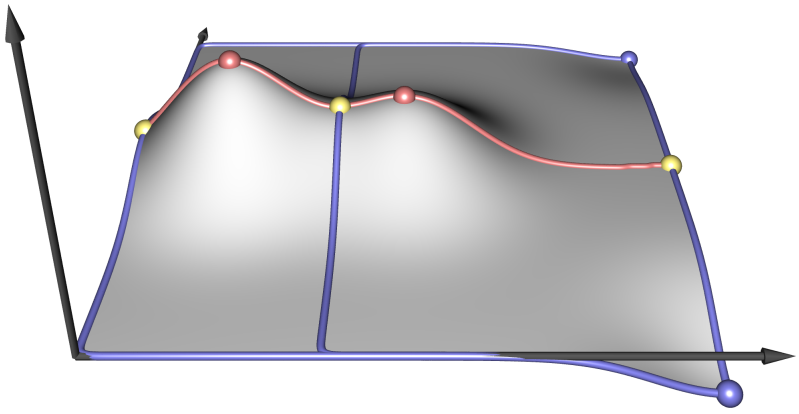


Figure 3.14 Morse-Smale complex. An illustration of the Morse-Smale complex of the smooth function f which is also shown in Figure 3.13. The complex consists of maxima, saddles and minima (red, yellow and blue spheres), and separatrices (red and blue lines). It segments the domain into regions of uniform gradient flow and has been computed using the algorithms introduced in Chapter 4.

3.3.2 Discrete Morse functions

As mentioned at the end of Section 3.2.2, we will restrict all definitions to the simple case of simplicial complexes even though they are applicable to the more general case of a regular cell complex.

Let K denote a finite d-dimensional simplicial complex and let f denote a function that assigns each simplex in K a real number. We now define a special boundary operator $\partial_f : K \to \mathcal{P}(K)$ by

$$\partial_f \alpha = \{\beta \in \partial\alpha : \ f(\beta) \geq f(\alpha)\}.$$

A function $f : K \to \mathbb{R}$ is now called a ***discrete Morse function*** if for every $\alpha \in K$ we have $|\partial_f \alpha| \leq 1$ and $|\partial_f^{-1}\alpha| = |\{\beta : \ \alpha \in \partial_f \beta\}| \leq 1$. Examples of discrete Morse functions are shown in Figure 3.15.

If α is a k-simplex with $\partial_f \alpha = \partial_f^{-1}\alpha = \emptyset$, then α is called a ***critical simplex*** of index k. If $d = 2$ we will call a critical simplex of index 2 a maximum, of index 1 a saddle, and of index 0 a minimum. Examples of discrete Morse functions containing critical simplices are shown in Figure 3.15.

Let m_k denote the number of critical simplices with index k of a discrete Morse function f defined on a d-dimensional simplicial complex. A direct consequence (see Section 3.3.3) of the definition of a discrete Morse function is that the ***Poincaré-Hopf equality***

$$m_0 - m_1 + \ldots + (-1)^d m_d = \chi(K), \tag{3.7}$$

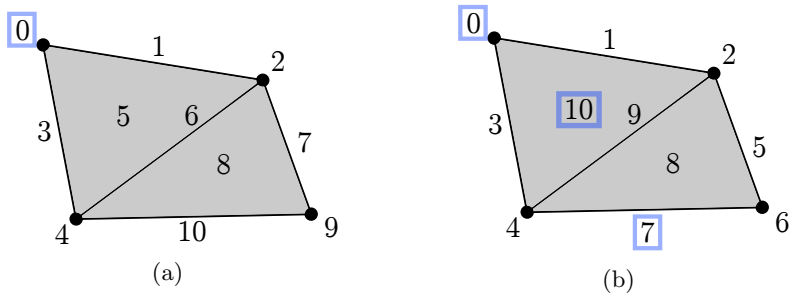


Figure 3.15 Discrete Morse functions. (a)-(b) a simplicial complex K with two functions $f : K \to \mathbb{R}$ that satisfy the constraints of a discrete Morse function. (a) the discrete Morse function contains a single critical simplex of index 0 (blue square). (b) the discrete Morse function contains three critical simplices of index 0, 1 and 2 (blue squares).

where $\chi(K)$ denotes the Euler characteristic (3.5) of K, is always satisfied. The alternating sum of critical points is thereby a topological invariant. For example, if K is a torus, i.e. $\chi(K) = 0$, then the number of saddles of any discrete Morse function f must always equal the sum of its maxima and minima.

A more fundamental result concerning the relation of the critical points of a discrete Morse function with the Betti numbers of the complex is proven in [For98b]. For any k the ***strong Morse inequality***

$$m_k - m_{k-1} + \ldots + (-1)^k m_0 \geq b_k - b_{k-1} + \ldots + (-1)^k b_0 \tag{3.8}$$

is always satisfied. For example, if f is a discrete Morse function defined on a torus ($b_0 = 1, b_1 = 2, b_2 = 1$), we know that there are always strictly more saddles than there are minima: $m_1 - m_0 \geq 2 - 1$.

Adding the strong Morse inequality for k to the strong Morse inequality for $k-1$ we obtain the ***weak Morse inequality***: $m_k \geq b_k$. For example, this implies that any discrete Morse function on a torus contains at least two saddles.

3.3.3 Morse homology with $\mathbb{Z}_2$ coefficients

To state the more general results by Forman we need the notion of a gradient vector field associated to a discrete Morse function f.

The set of pairs $V_f = \{\{\alpha, \beta\} : \{\beta\} = \partial_f \alpha\}$ is called the ***combinatorial gradient field*** associated to a Morse function f. A combinatorial gradient field can be intuitively visualized by drawing an arrow from β to α for each pair contained in V_f as depicted in Figure 3.16. Note that a simplex is critical iff it is not contained in any pair of V_f.

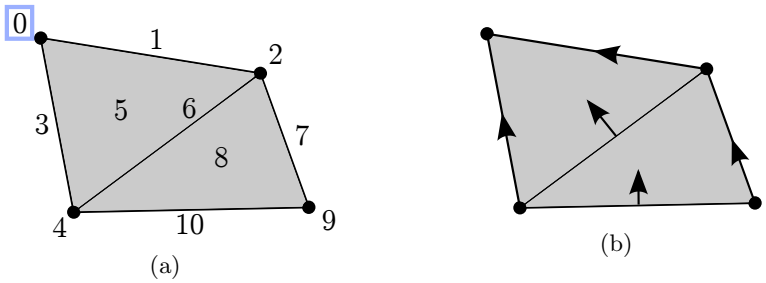


Figure 3.16 Combinatorial gradient field. A discrete Morse function (a) and its associated combinatorial gradient field (b). The presence of the critical simplex in (a) is indicated in (b) by being neither the start nor the end of an arrow.

Let $\alpha_0, \alpha_1, \ldots, \alpha_\ell$ denote $(k+1)$-simplices and let $\beta_0, \beta_1, \ldots, \beta_\ell$ denote k-simplices. The sequence $\alpha_0\beta_0\alpha_1\beta_1 \cdots \alpha_\ell\beta_\ell$ is now called a V_f-path of index k if $\beta_\ell \in \partial\alpha_\ell$ and $\{\beta_\ell, \alpha_{\ell+1}\} \in V_f$. If both α_0 and β_ℓ are critical, then a V_f-path is called a ***separatrix*** of index k. Examples of separatrices are depicted in Figure 3.17.

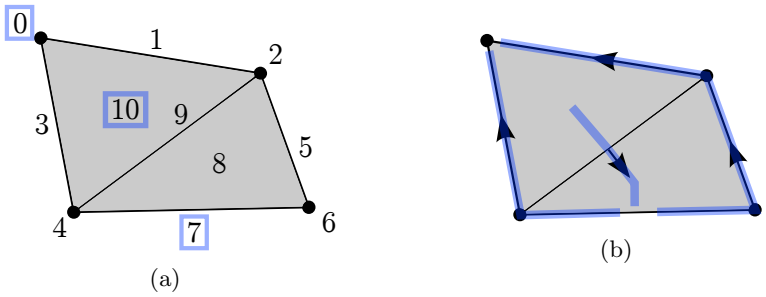


Figure 3.17 Separatrices. (a) a discrete Morse function with three critical simplices (blue squares). (b) the associated combinatorial gradient field contains three separatrices (blue lines) connecting the critical simplices.

The separatrices and critical points defined by a combinatorial gradient field V_f will now be used to define a chain complex associated with V_f. Let C_k^V denote the power set of the critical k-simplices. The boundary of a critical k-simplex is now defined as the set of critical $(k-1)$-simplices that are connected to it by an odd number of separatrices. The boundary $\partial_k^{V_f}$ of a set of critical k-simplices is now induced by the symmetric difference of the boundaries of the individual critical k-simplices. One can show that $\partial_{k-1}^{V_f}\partial_k^{V_f}\alpha = \emptyset$ for any critical k-simplex α. The sequence

$$C^{V_f}: \quad \emptyset \xrightarrow{\partial_{d+1}^{V_f}} C_d^{V_f} \xrightarrow{\partial_d^{V_f}} C_{d-1}^{V_f} \xrightarrow{\partial_{d-1}^{V_f}} \ldots \xrightarrow{\partial_1^{V_f}} C_0^{V_f} \xrightarrow{\partial_0^{V_f}} \emptyset \tag{3.9}$$

is therefore exact, and defines a chain complex, called ***Morse-Smale complex*** and we can use the same construction as in Section 3.2 to define the homology of this complex, called ***Morse homology*** with $\mathbb{Z}_2$ coefficients. One fundamental result of [For98b] is that the homology groups of C^{V_f} are isomorphic (i.e. in some sense equal) to the homology groups of C as defined in (3.4).

Note that in general the chain complex C^{V_f} does not have the structure of a simplicial complex. In two dimensions, a maximum may be connected to an arbitrary number of saddles while the boundary of a triangle always consists of three edges.

3.3.4 Combinatorial vector fields

One can prove that if α is not critical, then either $\partial_f \alpha = \emptyset$ or $\partial_f^{-1} \alpha = \emptyset$ which implies that V_f is always pairwise disjoint. This fact leads to the following generalization of combinatorial gradient fields. A pairwise disjoint set of pairs $V = \{\{\alpha, \beta\} : \beta \in \partial\alpha\}$ is called a ***combinatorial vector field*** [For98a]. In the context of combinatorial vector fields, a simplex $\alpha \in K$ is called a critical simplex if it is not contained in any element of V. Note that this condition is equivalent to the one in the preceding section when $V = V_f$ for some discrete Morse function f. The definition of paths and separatrices also carries over in a straightforward manner by replacing V_f with V in Section 3.3.3.

Using the notion of a combinatorial vector field it is very easy to verify that the Poincaré-Hopf equality (3.7) is always satisfied for any combinatorial vector field defined on a d-dimensional simplicial complex $K = \bigcup_{\ell=0}^{d} K_\ell$. We first recall the identity

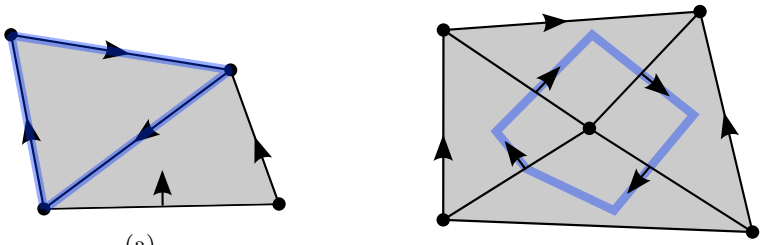
$$\chi(K) = |K_0| - |K_1| + \ldots + (-1)^d |K_d|,$$

mentioned in Section 3.2.2. We can now prove the Poincaré-Hopf equality by induction [For98a]:

1. The empty combinatorial vector field $V = \emptyset$ obviously satisfies (3.7) since every simplex is critical.

2. When we add an element to V, we remove two critical simplices whose dimension differs by one. This maintains (3.7).

To state the strong Morse inequalities for the combinational vector field case we need to define the notion of a closed V-path. Let $\alpha_0, \alpha_1, \ldots, \alpha_r$ denote $(k+1)$-simplices and let $\beta_0, \beta_1, \ldots, \beta_r$ denote k-simplices. The sequence $\alpha_0\beta_0\alpha_1\beta_1 \cdots \alpha_r\beta_r\alpha_0$ is now called a ***periodic orbit*** of index k if $\beta_\ell \in \partial\alpha_\ell$, $\{\beta_\ell, \alpha_{\ell+1}\} \in V$ and $\{\beta_r, \alpha_0\} \in V$. Examples of periodic orbits are depicted in Figure 3.18.



(a) (b)

Figure 3.18 Periodic orbits. (a) a combinatorial vector field defined on a simplicial complex containing a periodic orbit (blue) of index 0. (b) another combinatorial vector field containing a periodic orbit of index 1.

The strong Morse inequalities proven in [For98a] take on a particularly simple form in the case of a special class of combinatorial vector fields. A combinatorial vector field is called ***Morse-Smale*** [For98a] if the set of its periodic orbits is pairwise disjoint.

Let m_k denote the number of critical simplices with index k of a combinatorial Morse-Smale vector field V defined on a simplicial complex K and O_k the number of its periodic orbits with index k. The strong Morse inequalities then take on the simple form

$$O_k + m_k - m_{k-1} + \ldots + (-1)^k m_0 \geq b_k - b_{k-1} + \ldots + (-1)^k b_0, \qquad (3.10)$$

where b_k denotes the kth Betti number of K.

It is not easy to efficiently check whether a combinatorial vector field is of type of Morse-Smale since we would need to iterate all its periodic orbits. Fortunately, there is a large set of simplicial complexes, which are quite common in practice, where every combinatorial vector field is of type of Morse-Smale. We call a d-dimensional simplicial complex K ***manifold-like*** if each $(d-1)$-simplex α is a face of at most two simplices. In geometric terms this means that in two dimensions an edge can only be adjacent to at most two triangles. A common example of this type of complex is a triangulation of a surface. Figure 3.19 illustrates the definition of a manifold-like simplicial complex.

Using the graph theoretic formulation of discrete Morse theory which will be introduced in Section 4.1 it is easy to show that every combinatorial vector field defined on a two dimensional manifold-like simplicial complex is of type Morse-Smale and thereby satisfies (3.10). For example, a combinatorial vector field defined on a triangulation of a torus with no critical simplices therefore contains at least one periodic orbit of index 0 and one periodic orbit of index 1.

We say that a combinatorial vector field V is a combinatorial gradient field if there exists a discrete Morse function f with $V_f = V$. Chari shows in [Cha00] that f exists iff V does not contain any periodic orbits. This result implies

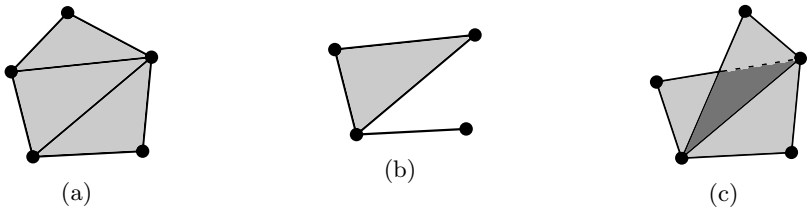


Figure 3.19 Manifold-like simplicial complex. (a)-(b) simplicial complexes that are manifold-like. (c) a simplicial complex which is not manifold-like.

that one can just as well define the notion of a combinatorial gradient field in terms of a combinatorial vector field. We will follow this approach in Chapter 4 since it encompasses both cases.

Chapter 4

Computational Discrete Morse Theory

In this chapter a unified framework for the extraction of extremal structures is presented. This framework is applicable to vector fields, scalar fields, divergence-free vector fields, and time-dependent scalar fields. We assume that these fields are defined on a regular cell complex, e.g. a triangulation or a quadrangulation. This makes our framework extremely flexible and most data sets encountered in practice can be directly analyzed.

All algorithms are only evaluated with two dimensional data sets. However, almost all concepts and algorithms are at least in part also applicable to higher dimensions, see e.g. [GRP$^+$] or [GRWH]. We will discuss potential obstructions and straightforward extensions in Chapter 5.

The theoretical foundation of this approach is given by discrete Morse theory as introduced in Section 3.3. This makes the computed extremal structures provably consistent with the topological invariants of the domain described in Section 3.2.

All definitions and algorithms are given in a simple graph theoretic notation introduced in Section 3.1. The advantage of this graph theoretic formulation is twofold. First, it allows to make use of existing graph theoretic algorithms such as breadth first search or the Hungarian method. This leverages a lot of existing knowledge concerning efficient algorithms and compact data structures. Secondly, it enables a short but concise description of the algorithms in pseudo code by exploiting the common familiarity of the reader with graph theoretic notions.

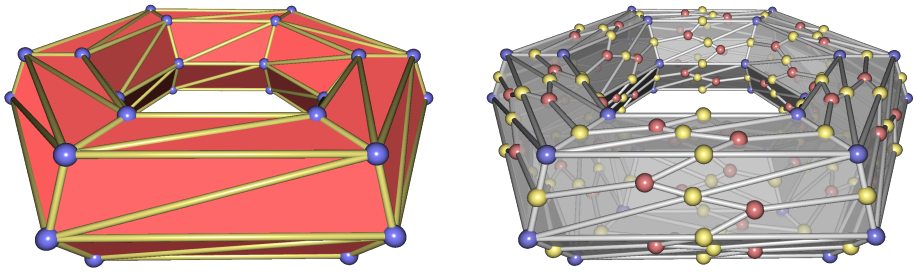
This chapter is structured as follows. Section 4.1 provides a graph theoretic formulation of discrete Morse theory and introduces the notation used throughout this chapter. Section 4.2 introduces and motivates our level-of-detail like hierarchical representation of the extremal structures. In Section 4.3 a generic algorithmic pipeline is presented that unifies the treatment of the different types of data. Sections 4.4 - 4.7 contain the data type specific algorithms for vector fields, scalar fields, divergence-free vector fields, and time-dependent scalar fields and evaluate their usefulness on various data sets.

This framework was implemented as a set of modules in the visualization and data analysis software Amira [SWH05]. It can be made available to researchers for evaluation purposes.

4.1 Graph theoretic formulation

We begin with a graph theoretic description of the domain of the data. We assume that the domain Ω is represented by a d-dimensional regular cell complex, as introduced in Section 3.2.2. All definitions thereby apply also directly to the simpler but maybe more familiar notion of a simplicial complex.

The basic idea is to construct a graph to represent the regular cell complex C in a combinatorial setting. The nodes of this graph represent the cells of C and the edges represent its boundary operator ∂. Furthermore, each node is labeled with the dimension of the cell that it represents and each edge is labeled with the minimum of the labels of its nodes. Such a graph of a cell complex that consists of a single triangle is shown in Figure 4.2a. A more complex example is shown Figure 4.1.



(a) (b)

Figure 4.1 Cell graph illustration. (a) a cell complex of a torus consisting of 0-cells (blue), 1-cells (yellow), and 2-cells (red). (b) the induced cell graph of the cell complex consists of nodes (spheres) that represent the cells and are colored according to the dimension, and edges (white) that represent the boundary relationship of the cells.

Formally, the ***cell graph*** associated to a cell complex $C = \bigcup_{\ell=0}^{d} C_\ell$, where C_k denotes the k-dimensional cells, is a labeled graph $G_C = (N, E, \varphi, \psi)$ with

- $N = \bigcup_{\ell=0}^{d} C_\ell$,
- $E = \{\{\alpha, \beta\} \subseteq N : \beta \in \partial\alpha\}$,
- $\varphi : N \to \{0, 1, \ldots, d\}$, $\varphi(\alpha) \mapsto k$, where $\alpha \in C_k$,
- $\psi : E \to \{0, 1, \ldots, d-1\}$, $\psi(\{\alpha, \beta\}) \mapsto \min(\varphi(\alpha), \varphi(\beta))$.

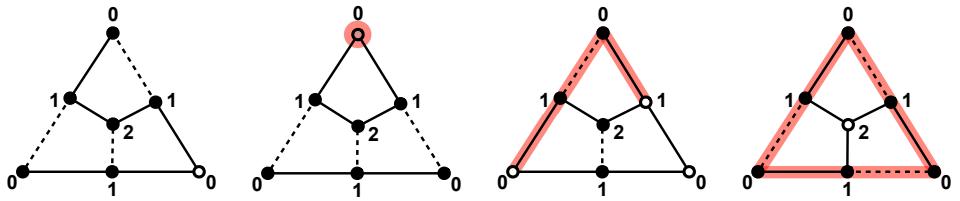


Figure 4.2 Basic definitions. (a) a combinatorial vector field (dashed) on the cell graph of a single triangle. The numbers correspond to the dimension of the represented cells, and matched nodes are drawn solid. (b) a critical point of index 0. (c) a 0-separatrix. (d) a 0-orbit.

To simplify notation, we will often indicate the label of a node φ using a superscript, i.e. a node v with $\varphi(v) = k$ is referred to by v^k. Similarly, we refer to an edge e with $\psi(e) = k$ by e^k. Note that this notion of a cell graph corresponds to the definition of the Hasse diagram given in [For01, Cha00].

The node labels φ and edge labels ψ supply G_C with a layered structure – a cell graph that represents a d-dimensional cell graph consists of $d-1$ layers. The ℓ^{th} layer represents the relationship of the cells in C_ℓ with the cells in $C_{\ell+1}$. Formally, each layer is a subgraph defined by

$$G_C^{k,k+1} = (\varphi^{-1}(k) \cup \varphi^{-1}(k+1), \psi^{-1}(k)). \tag{4.1}$$

The definitions of discrete Morse theory introduced in Section 3.3 will now be reformulated in the language of graph theory as introduced in Section 3.1. The main benefit of this reformulation is that the resulting definitions allow for a straightforward application of algorithms from graph theory for computational purposes.

Recall that a combinatorial vector field V on a cell complex C is defined as pairwise disjoint set of pairs $V = \{\{\alpha, \beta\} : \beta \in \partial\alpha\}$. A combinatorial vector field can thereby be equivalently defined as a matching M of the set of matchings $\mathcal{M}$ in a cell graph G_C. Simple examples illustrating this idea are shown in Figure 4.2.

A critical cell of index k is characterized in Section 3.3.3 as a k-cell which is not contained in any pair of the combinatorial vector field V. In graph theoretic terms, a critical cell α of index k is thereby an unmatched node α^k of a given matching in a cell graph. In the graph theoretic setting we will sometimes call a critical cell a critical node, or a critical point of the graph. The definition of a critical cell is illustrated in Figure 4.2b.

The graph theoretic formulation of the notion of a separatrix of index k of a combinatorial vector field is a little more involved. We first recall the definition of a V-path given in Section 3.3.3. Let $\alpha_0, \alpha_1, \ldots, \alpha_m$ denote $(k+1)$-cells and

let $\beta_0, \beta_1, \ldots, \beta_m$ denote k-cells. Then the sequence $\alpha_0\beta_0\alpha_1\beta_1 \cdots \alpha_m\beta_m$ is called a V-path of index k if $\beta_\ell \in \partial\alpha_\ell$ and $\{\beta_\ell, \alpha_{\ell+1}\} \in V$.

In the graph theoretic setting, a V-path of index k can thereby be defined as an alternating path in $G_C^{k,k+1}$ with respect to a matching defined on a cell graph G_C. We will also refer to a V-path of index k as a ***k-streamline*** of the combinatorial vector field.

A separatrix of index k is a V-path of index k whose start and end cell are critical. The graph theoretic analogue of separatrix of index k is therefore an augmenting path in $G_C^{k,k+1}$, called a ***k-separatrix*** of the combinatorial vector field. Similarly, a periodic orbit of index k can be defined as an alternating cycle in $G_C^{k,k+1}$ and is called a ***k-orbit*** of the combinatorial vector field. The definitions of a k-separatrix and a periodic orbit are illustrated in Figure 4.2c and Figure 4.2d.

Note that the rest of a 0-streamline onwards from a 0-cell β_ℓ is completely determined. First note that due to the definition of a matching, $\alpha_{\ell+1}$ is uniquely determined. And since the path has to be alternating and $|\partial\alpha_{\ell+1}| = 2$ the next 0-cell is also unique. Several 0-streamlines can therefore only merge. In the case of a manifold-like cell complex, see Section 3.3.4 for the definition, the same argument shows that $(d-1)$-streamlines can only split since any $(d-1)$-cell β is in the boundary of at most 2 d-cells. This implies that 0-orbits and $(d-1)$-orbits are always pairwise disjoint on a manifold-like cell complex. It follows directly that all combinatorial vector fields defined on a two dimensional manifold-like cell complex are therefore of type of Morse-Smale and (3.10) applies.

Recall that a combinatorial gradient field can be defined as a combinatorial vector field that does not contain any periodic orbits. The graph theoretic representation of a combinatorial gradient field is therefore given by a matching in a cell graph G_C that does not contain any alternating cycle in any subgraph $G_C^{\ell,\ell+1}$. Such a matching is also called a ***Morse matching*** [JP06]. We will denote the set all Morse matchings by $\mathcal{M}^\phi$. Examples of Morse matchings are shown in Figure 4.2abc whereas the matching in Figure 4.2d is not a Morse matchings.

In the remainder of this chapter we will make use of the conceptual and notational frameworks from graph theory (matchings and unmatched nodes) as well as discrete Morse theory (combinatorial vector fields and critical simplices) depending on the given context.

4.2 Hierarchical combinatorial vector fields

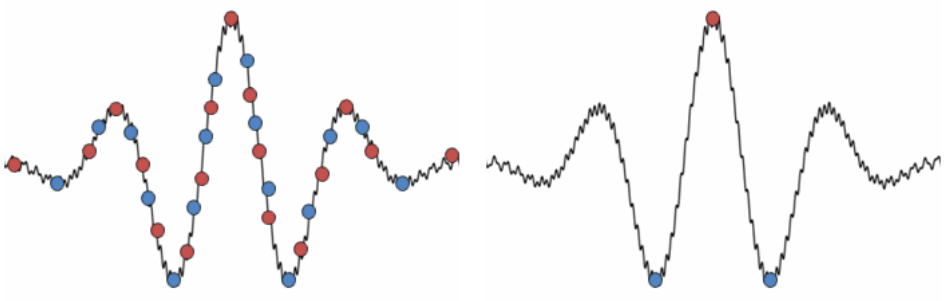
In many cases one is not interested in all the extremal structures of the given data set. This is especially true when the data contains noise. In such a case the number of critical points can be huge, and not much insight can be gained by extracting the extremal structure of the data. In some applications one is also rather interested in the dominant, or stable, extremal structures of the data set since only they correspond to the relevant features of the data.

To deal with this problem, we propose to compute a hierarchy of combinatorial vector fields

$$\mathcal{V} = (V_\ell)_{\ell=\ell_0,\dots,\ell_n} \text{ with } V_\ell \in \mathcal{M}, \tag{4.2}$$

to represent a given data set. To represent gradient or scalar fields we compute a sequence of combinatorial gradient fields, i.e. the set of matchings $\mathcal{M}$ is replaced by the set of Morse matchings $\mathcal{M}^\phi$ in the above definition.

The finest level of this hierarchy V_{ℓ_0} contains all critical points of the data set, while a coarser level V_{ℓ_k} only contains the important critical points of the data. See Figure 4.3 for an illustration of this idea.



(a) (b)

Figure 4.3 Extremal structures of noisy data. (a) a noisy scalar function. A fine level of the hierarchy of extremal structures contains a large number of minima (blue) and maxima (red). (b) a rather coarse level of the hierarchy of extremal structures contains only the dominant maxima and minima

Our hierarchy of combinatorial vector fields is motivated as follows:

1. The number of unmatched nodes of a matching M in a graph $G = (N, E)$ is given by $|N| - 2|M|$ since each edge contained in a matching covers two nodes. The number of critical nodes in a combinatorial vector field is thereby controlled by its size. The number of edges contained in combinatorial vector field should therefore increase when we traverse the hierarchy (4.2), i.e. $|V_{k+1}| = |V_k| + 1$

2. Let $c : \mathcal{M} \to N$ denote the set of critical nodes of a given combinatorial vector field. Since we want a hierarchical representation of the extremal structure, the critical nodes should be nested with respect to the sequence (4.2), i.e. $c(V_{k+1}) \subset c(V_k)$.

3. When the data set is large, the computation of the combinatorial vector fields can be costly and take a lot of time. Also, the relevant or interesting hierarchy level may a priori not be known. It is therefore beneficial if we can compute the hierarchy in a preprocessing step and then allow the user to iterate through sequence (4.2) interactively. To allow for this approach, the complete hierarchy needs to have a compact representation and we need to be able to switch between different hierarchy levels quickly.

To achieve the above goals we propose to compute a hierarchy that is based on augmenting paths. Let M denote a matching and p an augmenting path in this matching. As described in Section 3.1 $M \,\Delta\, p$ is then a matching whose size is increased by one and the unmatched nodes of $M \,\Delta\, p$ are contained in M. A depiction of this idea is shown in Figure 4.4.

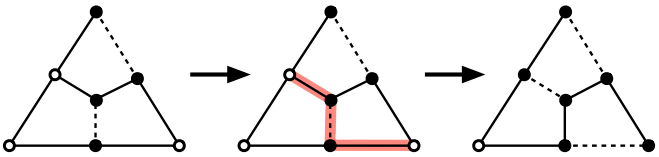


Figure 4.4 Illustration of the hierarchical representation of the extremal structure using augmenting paths. Given the heaviest matching with two edges (left) its heaviest augmenting path (middle) is computed. We augment the matching along this path to get the heaviest matching with three edges (right).

Recall that $\mathcal{A}_M$ denotes the set of augmenting paths in the matching M. A sequence of combinatorial vector fields (4.2) with

$$V_\ell \,\Delta\, V_{\ell+1} \in \mathcal{A}_{V_\ell}, \quad \ell = \ell_0, \ldots, \ell_{n-1}, \tag{4.3}$$

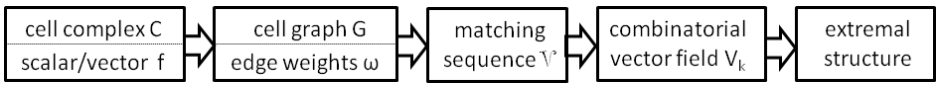
then satisfies the first two goals stated above.

In fact, it almost always satisfies also the third goal since to store the complete hierarchy we only need to store V_{ℓ_0} and all computed augmenting paths. In general an augmenting is not space filling and the number of paths corresponds to the number of hierarchy levels we computed. The sequence (4.2) therefore usually does not require much storage space and allows the user to browse through all hierarchy levels interactively.

4.3 Generic algorithmic pipeline

A significant advantage of the graph theoretic formulation of discrete Morse theory introduced in this chapter is the high level of abstraction of the geometric representation of the data set. In practice many different types of meshes are employed to discretize a certain domain. Some examples of the primitives that make up such meshes are triangles, quadrangles, tetrahedrons, or prisms. Also the functions defined on the mesh may consist of scalar fields, general vector fields, or divergence-free vector fields. This section proposes an algorithmic pipeline that unifies the algorithmic treatment of all these different types of data sets.

The general pipeline for the specific algorithmic approaches described in Sections 4.4 - 4.6 share the following computational pipeline:



In the remainder of this section we provide a description of the individual parts of this pipeline and refer to necessary specializations described in detail in Sections 4.4 - 4.6.

4.3.1 Input data

Our computational framework requires as input a d-dimensional finite regular cell complex and a scalar or vector valued function f. As mentioned above, the assumptions that the data consists of a cell complex covers almost all kinds of geometry encountered in practice. In the subsequent steps of the pipeline we will assume that f is defined on the 0-cells of the complex. Because we will later need data values on all cells, we extend f to the higher dimensional cells by taking the average value of the incident lower dimensional cells.

4.3.2 Weighted cell graph

Using the regular cell complex, we first construct its associated cell graph $G = (N, E)$ as defined in Section 4.1. A spatial embedding $c : N \to \mathbb{R}^n$ of G can be defined using the embedding of the cell complex in $\mathbb{R}^n$. The embedding of the nodes that represent higher dimensional cells is thereby computed by taking the average of the coordinates from the incident lower dimensional cells. Note that this propagation of the coordinates is only sensible for cell complexes that consist of convex cells – for more general geometries a more intricate propagation has to be applied to get a sensible spatial embedding of the graph.

Throughout this thesis we will assume that the cell graph is sparse, i.e. $O(|N|) = O(|E|)$, and state all complexities of the algorithm using this as-

sumption. Note that almost all meshes employed in practical numerical simulations fulfill this requirement. For example, it is common to design triangulations such that the minimum angle in the mesh is bounded away from zero. This immediately implies that the associated cell graph is sparse.

We now define the edge weights $\omega : E \to \mathbb{R}$ of this graph. Let $e^p = \{u^p, w^{p+1}\}$ denote an edge of the graph. If e^p is a matching edge it can be thought of as an arrow pointing from u^p to w^{p+1}. We therefore need to assign a large weight to e^p if such an arrow reflects the flow behavior of (the gradient of) f well. In this thesis, we propose to measure the tangential flow of (the gradient of) f along e^p to achieve this. Using Stokes' Theorem, the edge weight ω for scalar input data f is thereby defined by

$$\omega(e^p) = f(w^{p+1}) - f(u^p), \tag{4.4}$$

whereas in case of vector field data f (assuming linear interpolation), its edge weight is defined by

$$\omega(e^p) = \big(f(w^{p+1}) + f(u^p)\big) \cdot \big(c(w^{p+1}) - c(u^p)\big)\,/2. \tag{4.5}$$

4.3.3 Matching sequence

We now define the optimization problem that results in a meaningful combinatorial representative of our input data f. Large values of the edge weights ω indicate that the flow behavior of the (gradient) vector field is represented well by that edge. A matching in the cell graph with high weight therefore represents the input data well. We therefore propose to compute a combinatorial vector field to represent f by finding the maximum weight matching in G

$$V = \arg\max_{M \in \mathcal{M}} \omega(M). \tag{4.6}$$

If we deal with scalar data, i.e. we need to compute a combinatorial gradient vector field V^ϕ, we simply replace the set of matchings $\mathcal{M}$ by the set of Morse matchings $\mathcal{M}^\phi$. Note that this restriction of the admissible matchings makes (4.6) an NP-hard problem in general [JP06].

Due to the matching property, the number of critical points is given by $|N| - 2\,|V|$. We can therefore compute a combinatorial vector field with a prescribed number of critical points by computing

$$V_k = \arg\max_{M \in \mathcal{M},\, |M| = k} \omega(M). \tag{4.7}$$

Let $k_0 = |V|$ denote the size of the maximum weight matching, and let $k_n = \max_{k \in \mathbb{N}} |V_k|$ denote the size of the heaviest maximum cardinality matching. From a data analysis point of view, V_{k_0} is a fine grained while V_{k_n} is the coarsest possible representation of the input data f. A hierarchy of combinatorial (gradient) vector fields $\mathcal{V}$ can now be defined as the the sequence of matchings

$$\mathcal{V} = (V_k)_{k=k_0,\dots,k_n} . \tag{4.8}$$

The main task of our computational framework is to compute the sequence (4.8) which is described in Sections 4.4 - 4.6.

Note that in the case of Morse matchings, the exact solution to (4.7) can in general not be represented as a sequence of augmenting paths: computing a maximum cardinality Morse matching is NP-hard [JP06] while computing augmenting paths can be done in polynomial time. Being able to represent the matching sequence as an initial matching and a sequence of augmenting paths is however essential to enable an interactive analysis of the hierarchical extremal structure, see Section 4.2.

We will therefore resort to an algorithm in Section 4.5 that only approximates (4.7) but results in a hierarchy that is represented by a sequence of augmenting paths.

4.3.4 Combinatorial (gradient) vector field

The initial matching V_{ℓ_0} and the sequence of augmenting paths (p_ℓ) computed in the preceding step allows for the reconstruction of an arbitrary element of the hierarchy of combinatorial vector fields. Each matching can be restored by iteratively taking the symmetric difference of V_{ℓ_0} with the augmenting paths: $V_{k+1} = V_k \,\Delta\, p_k$. Since this operation is quite fast in practice this enables the user to interactively select a combinatorial (gradient) vector field from the hierarchy with a prescribed number of critical points.

Alternatively, we can make use of the associated weight of each matching as an importance measure. The user can set a fraction $\theta \in [0,1]$ to select a combinatorial (gradient) vector field with a weight as close as possible to $\omega(V_{k_0}) + \theta\,(\omega(V_{k_n}) - \omega(V_{k_0}))$. This approach can be useful in dealing with noisy data. Noise induces a very complex extremal structure. The augmenting paths corresponding to the spurious extremal structure, however, have a very large weight. Setting θ to a small value therefore removes all spurious extremal structures while the dominant structure remains unchanged.

Note that the weight difference $\omega(V_j) - \omega(V_{j+1})$ gives us a measure for the importance of the simplification which is closely related to persistent homology [EHNP03]. For future reference, we call this difference of matching weights the *weight of cancellation j* and assign its value to the critical points it removes. If we are dealing with gradient vector field data, then this weight difference corresponds to the difference of the scalar values of the critical points being canceled. We can therefore make use of (4.8) to remove topological noise (see Figure 4.6), or to reduce the topological representation of the data to its dominant structures (see Figure 4.8). This enables a multi-scale topological analysis of vector fields.

4.3.5 Extremal structure

Given a combinatorial vector field, we can now extract its extremal structure – the critical points, separatrices, and periodic orbits.

The critical points can be easily extracted since they are simply the unmatched nodes u^p of the combinatorial vector field. The classification into sources, saddles, sinks (or in the scalar case minima, saddles, maxima) is given by the label p – it can be interpreted as the number of negative eigenvalues of the Hessian in the smooth case, see Section 3.3.2.

The computation of the separatrices is in general a lot more involved. The computation of the 0-separatrices is however always quite simple: we only need to iterate over the all saddles u^1 and compute the incident 0-separatrices. Since 0-streamlines can only merge (see Section 4.1), it suffices to follow the two alternating paths starting in u^1. When the cell complex is manifold-like the same procedure can be applied to compute the $(d-1)$-separatrices. If we stop this kind of line integration when we reach an already extracted separatrix the whole computation can be done in linear time – each node of the graph is only visited a constant number of times. Pseudo code for the extraction of the separatrices of a saddle in two dimensions is shown in Algorithm 12.

The efficient computation of the k-separatrices with $0 < k < d-1$ is however very challenging [GRP+].

In the vector case, we can also extract the periodic orbits. To compute the 0-orbits, we first iterate over all 1-nodes of the graph. Given a node u^1, we start the computation of the 0-streamlines that emanate at u^1. Each streamline is continued as long as the following node w^1 is not yet labeled, in which case it is labeled with u^1. If the label of w^1 equals u^1 we add w^1 to a set of seed points. We then iterate over all seed points and compute their combinatorial stream lines which yields all 0-orbits. Similar to the separatrix computation, the same procedure can be applied to compute all $(d-1)$-orbits if the cell complex is manifold-like. The computation of all 0-orbits and all $(d-1)$-orbits can be implemented in linear time using the above algorithm.

The efficient computation of the k-orbits with $0 < k < d-1$ is an open problem.

Note that in continuous vector field topology separatrices are sometimes defined differently - a streamline that connects a saddle with a periodic orbit is also called a separatrix. While this does not directly correspond to our definition of separatrices, we can also extract these lines by following the p-streamlines that emanate from a saddle.

4.4 Vector fields

This section introduces a purely combinatorial approach for the extraction of the extremal structure of a given vector field. This problem is directly suitable to the generic algorithmic pipeline described in Section 4.3. The only algorithm we have to describe is the computation of the sequence of maximum weighted matchings (4.7).

To do this one can use the Hungarian method introduced in Section 3.1 since a cell graph is always bipartite. A simple bipartition of the nodes $N = U \dot{\cup} W$ is given by a partition into the even and odd labeled nodes of the graph.

The main weakness of this approach is its computational complexity of $O(n^2 \log n)$, where n denotes the number of nodes in the data set. Even rather small datasets with 60k nodes take about 30 minutes to compute on workstation from 2008. The quadratic scaling in the runtime therefore severely limits the applicability to real world data sets - an application to large higher dimensional data sets also seems unfeasible using the Hungarian method.

We therefore propose a new algorithm in Section 4.4.1 that can replace the Hungarian method. This approximative algorithm has a lower empirical complexity of $O(n^{3/2} \log n)$. It reduces the runtime by several orders of magnitude for our data sets, while it produces visually the same results as the exact algorithm and preserves a certain monotonic behavior of the exact solution.

Almost all computational time of our algorithm is spent solving shortest path problems with negative weights. As this particular graph problem is easily parallelizable, we have implemented it in a massively parallel fashion using CUDA, which reduces the runtime even further.

There is a lot of literature on the approximation of the maximum weighted matching problem (4.6), see [HD04] for an overview. It is, however, unclear how one would efficiently extend these algorithms to compute the sequence of maximal weight matchings (4.7). Also, the graphs described in Section 4.1 have a very specific structure in their connectedness and weights that can be exploited by a custom algorithm.

The weight of the maximum weight augmenting path decreases as we iteratively compute (4.8), see [Sch03]. In the following approximation algorithm we make sure that this monotonic behavior is preserved.

Note that the monotonicity property of our approximation is necessary to maintain the relation of our hierarchy to the one computed using persistence [ELZ02]. If we deal with a gradient vector field, then the critical points should be canceled in an order determined by the difference of their scalar values. The exact algorithm always cancels the pair of connected critical points with the smallest scalar difference. Our monotony preserving approximation algorithm guarantees that the scalar difference of canceled pairs always increases as we simplify the topological skeleton. As can be seen in Figure 4.6,

this property is not only necessary to maintain the relation to persistence, but it also seems to be sufficient in practice.

4.4.1 Algorithm

The Hungarian method has a computational complexity of $O(n^2 \log n)$, as it needs to solve n shortest path problems with positive weights. To reduce this rather large runtime we propose to approximate the exact solution by augmenting the matching along *all* shortest paths computed when we solve the shortest path problem. As demonstrated in Section 4.4.2 this leads to a significantly reduced overall runtime. More details on the algorithmic realization of this idea can be found below and the pseudo code is given in Algorithm 5. We call this step of our algorithm the *Predictor* phase.

If we continued in this fashion we would generate an approximation of (4.8) without the monotonic behavior mentioned at the end of Section 4.4. In practice this may result in a bad approximation (see Figure 4.6, bottom-left) of the exact solution (see Figure 4.6, top-left). We therefore maintain the monotonicity of the exact solution which leads to a good approximation (see Figure 4.6, top-right). To achieve this, we apply a roll-back operation to the matching sequence which guarantees the monotonic behavior as follows. After each Predictor phase (see above) of our algorithm, we compute the weight of the heaviest augmenting path of the current matching V_j. If this weight is smaller than the weight of the augmenting path that led to V_j we can continue with the Predictor phase. Otherwise we need to find a matching V_ℓ, $\ell < j$ where this property is fulfilled.

The algorithmic realization of this idea is described below and pseudo code is given in Algorithm 6. For future reference we call this step of our algorithm the *Corrector* phase.

For the overall combinatorial predictor-corrector algorithm that produces a monotone approximation of (4.8) we refer to the pseudo code shown in Algorithm 4. A simple edge weighted graph is used in Figure 4.5 to demonstrate our predictor-corrector algorithm.

We do not give a detailed computational complexity analysis of our algorithm. It is clear that it is polynomial, but calculating a worst case upper bound on the degree is not beneficial since it would be overly pessimistic for such a predictor-corrector algorithm. We can however comment on the empirical complexity for our test data sets: Section 4.4.2 indicates that our algorithm has a complexity of $O(n^{3/2} \log n)$ and we can provide some arguments why this can be expected.

As we augment the matching by all available augmenting paths with each application of Bellman-Ford, the number of uncovered nodes $S \setminus S(V_j)$ decreases exponentially with respect to the number of shortest path computations. We therefore only need to compute Bellman-Ford $O(\log n)$ often. Let

t denote the length of the longest shortest path in an edge weighted simplicial graph. Then the computational complexity of Bellman-Ford is given by $O(t\,n)$. Assuming a two dimensional manifold-like cell graph, the length of the longest shortest path is bounded in practice by $\sqrt{n}$. The complexity of Bellman-Ford for this class of graphs can therefore be estimated by $O(n^{3/2})$. The total complexity for our approximative algorithm is hence expected to be $O(n^{3/2}\log n)$, and this conjecture is substantiated in Table 4.1.

Algorithmic Details

We now proceed by giving a detailed and accurate description of our approximation algorithm for (4.8) motivated in above. To ensure a good reproducibility of our results this part will be quite technical. The main algorithm is given in Algorithm 4. The input of this algorithm is the edge weighted simplicial graph $G = (U\dot{\cup}W, L, \omega)$, where $U\dot{\cup}W$ denotes a bipartition of the nodes S of the graph. For a detailed description of this input data we refer to Section 4.1.

The output of Algorithm 4 consists of an approximation of the heaviest maximum cardinality matching V_{k_n} and an array of maximum weight augmenting paths P. V_{k_n} and P enable us to efficiently reconstruct all matchings in the sequence (4.8). This is due to the fact the last path stored in P is an alternating path in V_{k_n} whose endpoints are both matched. We can therefore reconstruct V_{k_n-1} by taking the symmetric difference $\triangle$ of V_{k_n} with the last path stored in P. This operation can be interpreted as an inverse augmentation of the matching. Storing the sequence (4.8) in this fashion is much more efficient than storing the individual matchings. In two dimensions, the

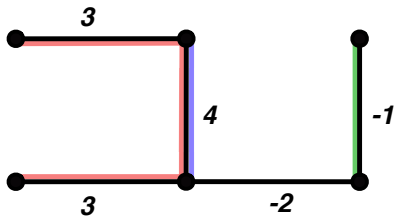


Figure 4.5 Comparison of algorithms for (4.8). A simple edge weighted graph illustrates the differences between the presented algorithms. The order of augmenting paths taken by the (exact) Hungarian algorithm is (blue, red, green) with corresponding weights (4, 2, -1). The approximate algorithm without the corrector phase chooses (blue, green, red) with corresponding weights (4, -1, 2). The full predictor-corrector algorithm works as follows: (blue, green, undo green, red, green) resulting in the monotone sequence of augmenting path weights (4, 2, -1).

size of an augmenting path is typically in $O(\sqrt{n})$ for our type of graphs (they are typically not space filling curves) while the size of a single matching is in $O(n)$.

Algorithm 4 Predictor-Corrector algorithm for (4.8)

Input: G
Output: V_{k_n}, P
1: M $\leftarrow \emptyset$
2: P $\leftarrow \emptyset$
3: **loop**
4: (isFinished, M, P) $\leftarrow$ Predict(G, M, P)
5: **if** isFinished = *false* **then**
6: (M, P) $\leftarrow$ Correct(G, M, P)
7: **else**
8: $V_{k_n} \leftarrow$ M
9: **return** (V_{k_n}, P)

We now describe the Predictor phase (Line 4 of Algorithm 4). The pseudo code for this procedure is given in Algorithm 5. Line 1 calls Algorithm 7 to compute for each node the heaviest augmenting path ending in it. The paths are stored implicitly via the node attribute *.predLink*, while the weight of the augmenting path is stored in the node attribute *.distance*. For a detailed description of Algorithm 7 see below. Line 2 computes the subset A of the nodes of partition W that are not covered by the matching and whose computed distance is finite. Line 8 extracts the augmenting path p that starts in the last element of A by following the node attribute *.predLink* through the graph and then removes the last element of A. Lines 9-12 check if p is still a valid augmenting path in the graph (the first augmenting path is always valid but may invalidate subsequent paths), augment the matching M along p by computing their symmetric difference $\triangle$ (see Figure 4.4) and append p to the list of augmenting paths P.

The Corrector phase (Line 6 of Algorithm 4) is similar to the Predictor phase described above. Its pseudo code is given in Algorithm 6. The procedure *weight* called in Lines 6,7 and 10 computes the weight of a given augmenting path, i.e. the alternating sum of edge weights.

Almost all computational time of the overall approximation algorithm is spent in Algorithm 7. We therefore present its pseudo code and some details on the application specific changes we have introduced compared to the standard Bellman-Ford shortest path algorithm. The input of Algorithm 7 consists of the edge weighted simplicial graph $G = (U \dot{\cup} W, L, \omega)$ and the current matching M. The output consists of the weight of the heaviest augmenting path for each node in the graph stored in *.distance*, and the respective augmenting paths stored implicitly in *.predLink*.

Algorithm 5 Predictor phase

Input: G, M, P
Output: isFinished, M, P

```
 1: (S.distance, S.predLink) ← BellmannFord(G, M)
 2: A ← {s ∈ W \ S(M): s.distance < ∞}
 3: A ← sortByDistance(A)
 4: if A = ∅ then
 5:   return (true, M, P)
 6: else
 7:   while A ≠ ∅ do
 8:     p ← getAugmentingPath(S, A.pop())
 9:     if isValidAugmentingPath(M, p) then
10:       M ← M △ p
11:       P.push(p)
12:   return (false, M, P)
```

Algorithm 6 Corrector phase

Input: G, M, P
Output: M, P

```
 1: loop
 2:   (S.distance, S.predLink) ← BellmannFord(G, M)
 3:   A ← {s ∈ W \ S(M): s.distance < ∞}
 4:   A ← sortByDistance(A)
 5:   p ← getAugmentingPath(S, A.top())
 6:   barrier ← weight(G, p)
 7:   if barrier ≤ weight(G, P.top()) then
 8:     return (M, P)
 9:   else
10:     while barrier > weight(G, P.top()) do
11:       M ← M △ P.pop()
```

Lines 1 - 16 initialize all variables so that the main Bellman-Ford loop computes the output described above. Note that Lines 9 - 15 correspond to the construction of the directed graph D_M described in Section 3.1. Lines 17 - 28 are a variant of Bellman-Ford optimized for the particular class of graphs we deal with and modified to support an efficient parallelization, see below. Instead of the *while* loop with an abort criterion one typically iterates Lines 22 - 25 n times. In the two dimensional case, the longest shortest path is typically in $O(\sqrt{n})$. It is therefore very beneficial to check whether we can abort this loop early. This is the purpose of Lines 16 - 18 and 28. Since the graph is not dense, only a small subset of nodes have to be considered in each iteration, which is achieved by the Lines 3, 20, 21 and 27.

Algorithm 7 Bellman-Ford matching variant

Input: G, M
Output: S.distance, S.predLink

```
1: for all s ∈ S do
2:    s.predLink ← nil
3:    isActive[s] ← true
4:    if s ∈ U \ S(M) then
5:       s.distance ← 0
6:    else
7:       s.distance ← ∞
8: L⃗ ← ∅
9: for all ℓ = {u, w} ∈ L, u ∈ U, w ∈ W do
10:   if ℓ ∈ M then
11:      L⃗ ← L⃗ ∪ (u, w)
12:      ω⃗(ℓ) ← ω(ℓ)
13:   else
14:      L⃗ ← L⃗ ∪ (w, u)
15:      ω⃗(ℓ) ← −ω(ℓ)
16: abort ← false
17: while abort = false do
18:   abort ← true
19:   for all s ∈ S do
20:      if isActive[s] = true then
21:         isActive[s] ← false
22:         for all ℓ = (u, s) ∈ L⃗ do
23:            if s.distance > u.distance + ω⃗(ℓ) then
24:               s.distance ← u.distance + ω⃗(ℓ)
25:               s.predLink ← ℓ
26:               for all ℓ = (s, w) ∈ L⃗ do
27:                  isActive[w] ← true
28:                  abort ← false
```

Parallelization

As practically all computational time of Algorithm 4 is spent in the Bellman-Ford algorithm (Lines 16-28 of Algorithm 7), its efficiency is critical to the overall runtime. In [HN07] it is shown that an implementation of this shortest path problem using CUDA can result in a great performance increase. Unfortunately we cannot directly make use of this parallel approach to Bellman-Ford, as we need to know not only the distances, but also the shortest paths themselves in our application. Including the computation of the shortest paths via the variable *.predLink* in the algorithm presented in [HN07] results in a race condition that leads to invalid results.

We therefore need to formulate Bellman-Ford in a parallel fashion such that there are no race conditions. This can be achieved by iterating over the incoming edges (Line 22 of Algorithm 7) for each node of the graph instead of its outgoing edges as in [HN07]. We can then parallelize the loop in Line 19 of Algorithm 7 as the write accesses are exclusive for each thread (the Boolean array *isActive* and the Boolean variable *abort* may be written to concurrently but this does not pose a problem).

4.4.2 Evaluation

This section evaluates the robustness, the approximation quality, and the performance of our approximation algorithm. To demonstrate its usefulness we also applied it to two real-world data sets and compare to a continuous approach to vector field topology.

Robustness and approximation quality

To determine the quality of our approximation algorithm we applied it to a synthetic data set shown in Figure 4.6. The data set was produced by sampling the analytic function $f : [-1, 1]^2 \to \mathbb{R}$

$$f(x, y) = \sin(10\,x)\,\sin(10\,y)\,e^{-3\,(x^2+y^2)} \tag{4.9}$$

on a uniform triangulation with $16k$ vertices, adding uniform noise of the range $[-0.05, 0.05]$ to the sub domain $[0, 1] \times [-1, 1]$ and taking the gradient. Note that the spatially varying amount of noise provides a special challenge for the topological analysis. Figure 4.6 shows a visualization of this triangulated vector field as a surface line-integral-convolution [MKFI97] (LIC) using the scalar value of f as the z-coordinate. We then solved (4.8) for this data set using the (exact) Hungarian method described in Section 3.1 and our new predictor-corrector algorithm described in this section. To evaluate the importance of the corrector phase, we also computed (4.8) using only the predictor phase. The 63 most important critical points, i.e. the critical points of V_{k_n-31}, are depicted for each algorithm in Figure 4.6.

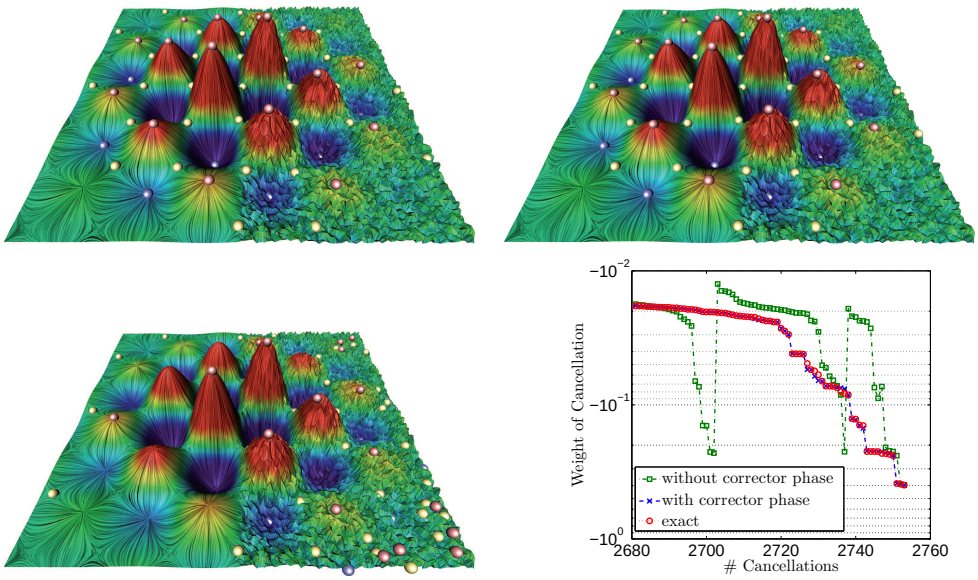


Figure 4.6 Approximation vs. exact method. A synthetic data set, described in Section 4.4.2, is analyzed using the exact method (top-left), our predictor-corrector approximation algorithm (top-right), and the predictor-only variant (bottom-left). The 63 most dominant critical points are shown as blue (sinks), yellow (saddles), and red (sources) balls. The weights of the last cancellations are shown for the different algorithms in a semi-logarithmic plot (bottom-right).

The result of our predictor-corrector approximation algorithm (top-right) is very similar to the exact result (top-left). The predictor-only result (bottom-left) however is quite different from the exact result. This bad approximation behavior of the predictor-only algorithm can also be seen in the weight of the cancellations depicted in Figure 4.6, bottom-right. The x-axis represents the pair cancellations, while the y-axis shows the weight of the cancellations. The proposed predictor-corrector algorithm (blue curve) closely follows the exact algorithm (red curve) and is monotonically decreasing. In contrast, the predictor-only algorithm (green curve) is quite different from the exact algorithm and does not preserve the monotonic behavior of the exact solution.

Applications

Figure 4.7 depicts a surface velocity field of a simulation of blood flow through a cerebral aneurysm done by the Biofluid Mechanics Lab of the Charité - Universitätsmedizin Berlin [CCA+05]. The cell graph of the triangulation consists of $60k$ nodes. The runtime for the computation of (4.8) using a simple implementation of the Hungarian method using a Dijkstras shortest path algorithm with a Fibonacci heap was about 30 minutes on workstation

from 2007.

The critical points in this vector field are stagnation points and thus of interest for the flow analysis. Our algorithm delivers a hierarchy of extremal structures which captures the dominant nature of the flow (see Figure 4.7). The blood enters the aneurysm at the bottom, and leaves it horizontally. This behavior is found by our algorithm and the global separation on the surface is extracted (see Figure 4.7c). This reduced flow structure may serve as a basis when comparing different cerebral aneurysms.

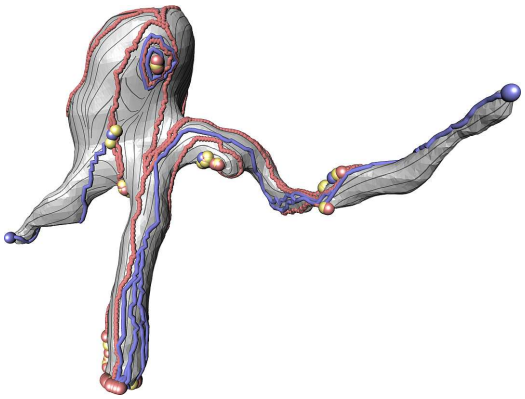
To demonstrate the usefulness of our algorithm we applied it to a real-world data set from climate research. This data set is a short subset of the IPCC AR4 climate projections, which were carried out at DKRZ by the Max-Planck-Institute for Meteorology with the coupled atmosphere-ocean model ECHAM5/MPI-OM. We have used the 10 meter wind components depicted by a surface LIC representation in Figure 4.8 using the pressure for the color values. We sampled this data set on a simplicial graph with about 2.5 million nodes and solved (4.8) with our combinatorial predictor-corrector algorithm. The runtime using a CUDA implementation of our algorithm was 4 minutes. The estimated runtime using the exact Hungarian method described in Section 3.1 is about 6 weeks.

The full set of critical points of the initial combinatorial vector field V_{k_0} without any simplification is shown in Figure 4.8. The critical points in this figure are scaled by their importance value given by the difference of the matching weights $\omega(V_j) - \omega(V_{j+1})$, see Section 4.3.4. Note that larger critical points correspond to strong pressure systems, even though the pressure values were not employed in the calculation of (4.8). This indicates the physical relevance of the persistence-like important measure induced by (4.8) for real-world vector field data.

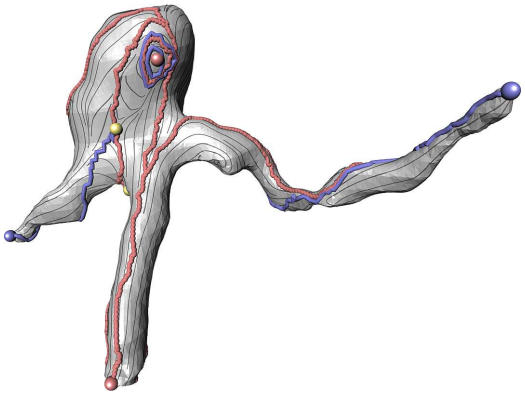
Comparison

To compare the presented combinatorial approach to vector field topology with a continuous one [Wei08], we analyzed the climate data set described above with both approaches. To compare the hierarchy of combinatorial vector fields to the single continuous extraction result, we selected the combinatorial vector field with the same number of critical points as the continuous extraction result. Both sets of critical points are depicted in Figure 4.9, the critical points extracted by the continuous method shown as black balls, the critical points computed by the combinatorial algorithm as white balls. Note that Figure 4.8 only seems to contain fewer critical points as they are scaled by their importance resulting in critical points that are smaller than a pixel.

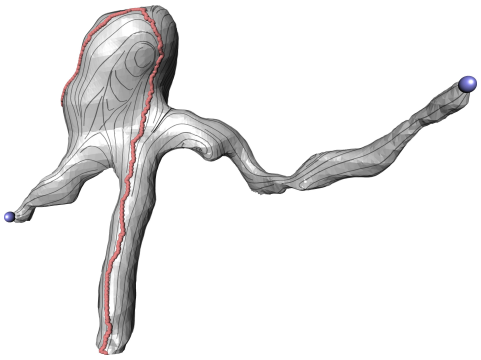
As can be seen in Figure 4.9, most critical points of both methods coincide. The critical points found by the continuous method, that are not included in the combinatorial result, all appear in flat regions of the vector field, i.e.



(a)



(b)



(c)

Figure 4.7 Vector field from biofluid mechanics. The vector field is visualized using the streamline seeding technique described in [RPP+09]. The extremal structures of a) V_{k_0}, b) V_{k_n-4} and c) V_{k_n} are shown. Sinks, saddles and sources are depicted as blue, yellow and red spheres. 0-separatrices and attracting periodic orbits are depicted as blue lines, while 1-separatrices and repelling periodic orbits are shown as red lines.

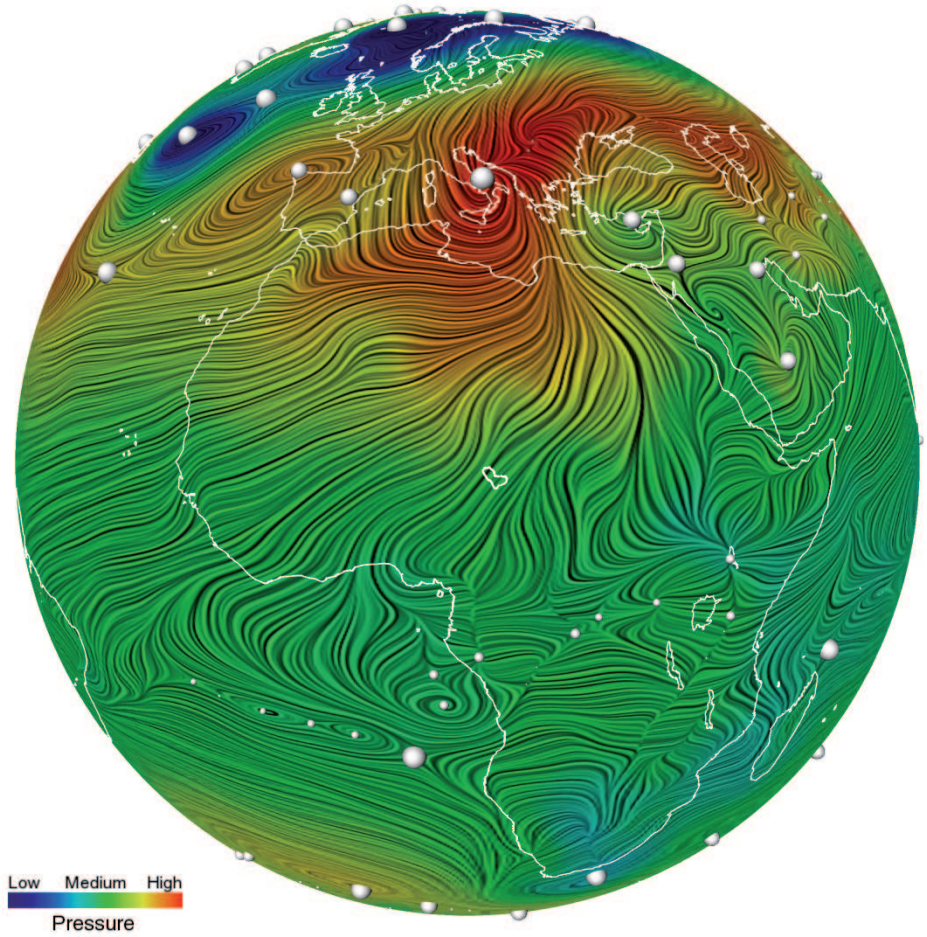


Figure 4.8 Importance measure illustration. The 10 meter wind components form a data set from climate research are depicted as a surface LIC using the pressure for the color. The white balls show the critical points computed by the combinatorial method. The size of the balls is determined by the persistence like importance measure described in Section 4.3.4. Note that the pressure is only depicted here to illustrate the physical relevance of the importance measure.

regions where the magnitude is close to zero. They are therefore not stable w.r.t. to perturbations of the data and may be considered as noise artifacts. Also, they may strongly depend on the chosen interpolation.

To describe the differences of our simplification strategy with the existing continuous extraction algorithms we can make use of the result shown in Figure 4.6. All existing simplification methods mentioned in Section 2.1 do not make use of the magnitude of the vector field. Therefore the peaks in the center of the data set would be given the same importance as the small hill tops

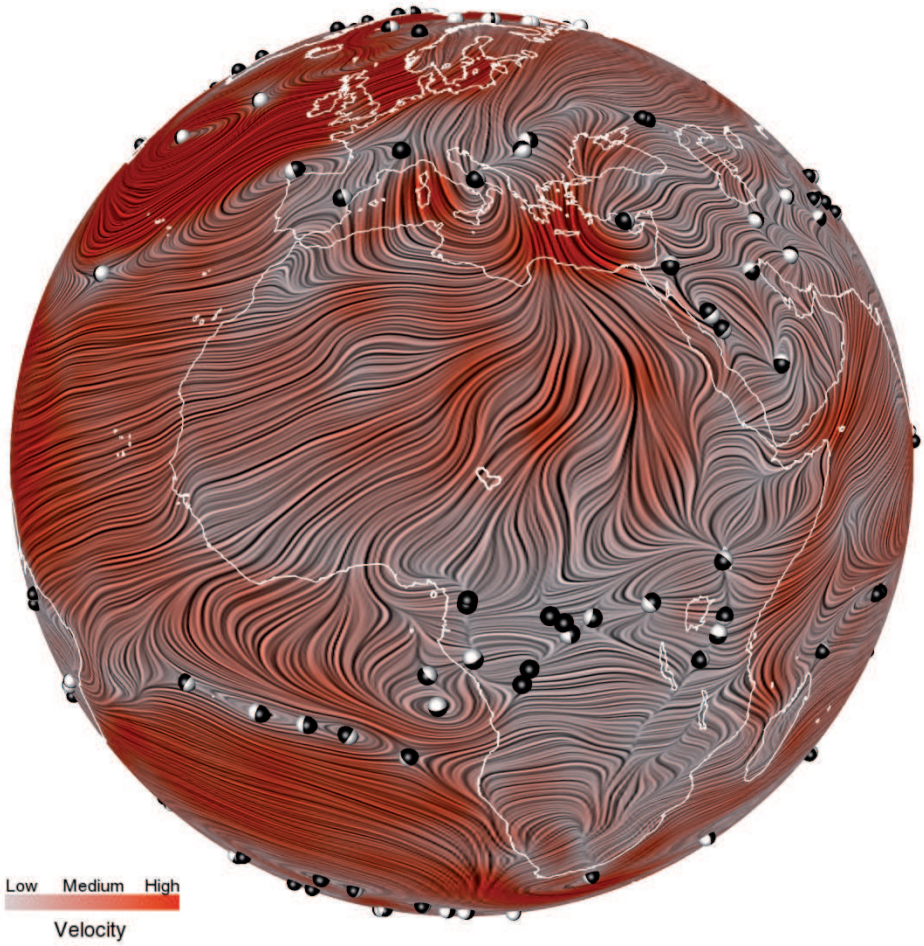


Figure 4.9 Comparison with continuous approach. The 10 meter wind components form a data set from climate research are depicted as a surface LIC using the wind velocity for the color. The black balls show the critical points of the vector field computed by a continuous method, while the white balls show an extraction result of the presented combinatorial approach.

near the boundary of the data set. In contrast, the presented combinatorial approach takes the magnitude of the vector field into account, which results in the simplification hierarchy shown in Figure 4.6, top. The small hill tops near the boundary of the data set are canceled at an early stage, while the peaks in the center are canceled last.

Performance

To measure the performance advantage of our algorithm over the exact algorithm we computed (4.8) for the aneurysm data set, the synthetic data

set (4.9), and four resolutions of a real-world data set from climate research using the exact Hungarian method described in Section 3.1 and our approximation algorithm. The timings for an Intel Core 2 Duo 3 GHz CPU with a Nvidia Geforce GTX 260 Core 216 graphics card are given in Table 4.1. To determine the parallel scalability we computed all data sets with 1 thread, 2 threads, and on the GPU. The CPU version was implemented with OpenMP, while the GPU version was implemented in CUDA. Due to the extremely long runtime of the exact method we have not measured its runtime for the larger data sets.

In theory, Algorithm 4 could have a very large computational complexity - even an upper bound for the complexity is hard to derive due to the predictor-corrector interplay. In practice however, our approximation algorithm has an empirical complexity of $O(n^{3/2} \log n)$ as can be seen in the 1×CPU column of Table 4.1. The OpenMP implementation shows a near perfect parallel scaling when going from one thread to two threads. The speed up provided by the CUDA implementation ranges between 1.5× and 30× - the larger the data set the bigger the speed up.

Table 4.1 Runtime analysis. (4.8) was computed for various data sets with the exact Hungarian method and the approximative algorithm using an OpenMP implementation (CPU) and a CUDA implementation (GPU).

Name	#nodes	exact	1×CPU	2×CPU	GPU
Aneurysm	60k	2360s	12s	6s	4s
Synthetic	97k	6258s	31s	16s	8s
Climate 1	154k	15919s	55s	28s	10s
Climate 2	614k	*	569s	284s	52s
Climate 3	2458k	*	4982s	2619s	237s
Climate 4	9830k	*	37602s	19014s	1220s

4.4.3 Discussion

As demonstrated in Section 4.4.2 the hierarchy of the extremal structure is maintained by our approximation algorithm. Each point is given an importance value that indicates its relevance in the overall data set. Since the approximation algorithm results in a matching sequence, the resulting topological skeletons are always consistent with the topology of the domain. Also, our predictor-corrector algorithm does not contain any computational parameters.

The main weakness of exact method for the computation of (4.8) is its large computational runtime. This weakness is alleviated by our algorithm that reduces the runtime from weeks to minutes for large data sets, see Table

4.1. While this algorithm is only approximative in nature, it produces results that are visually indistinguishable from the exact solution of (4.8), see Section 4.4.2. The algorithm is also capable of dealing with data sets with varying amounts of noise, as can be seen in Figure 4.6.

Our predictor-corrector approximation algorithm for (4.8) produces results that are very close to the exact solution for the graphs we consider. We assume that this is due to the preservation of the monotony and the special structure of the cell graph and the symmetries of the edge weights (4.5). From a graph theoretic point of view it would be interesting to find out whether there are other classes of graphs where our algorithm for (4.8) produces such good results.

A possible limitation for the application of our algorithm to other graph problems is the fact that the Bellman-Ford algorithm does not work when the graph contains a circle with negative weight [Sch03]. In the exact method we can use Bellman-Ford due to the maximality property (4.7) of the computed matchings (see Section 3.1). In our approximate setting this maximality property does not necessarily hold and there may theoretically exist circles of negative weight. However in all of our experiments we have never encountered such a case. This may be due to the monotonic behavior of our approximation or the manifold structure of the graphs. A thorough theoretical investigation of this empirical observation may be worthwhile.

4.5 Gradient fields

This section introduces a purely combinatorial approach for the extraction of the extremal structure of a given scalar (or gradient) field. This problem is directly suitable to the generic algorithmic pipeline described in Section 4.3. The only algorithm we have to describe is the computation of the sequence of maximum weighted matchings (4.7) for the case of Morse matchings $\mathcal{M}^\phi$

In contrast to the vector field case, the computation of (4.8) for the scalar field case $\mathcal{M}^\phi$ is a lot more involved – in general it is NP-hard [JP06]. We therefore propose a simple approximation algorithm for this problem that produces a sequence of augmenting paths. This is essential for a compact representation of the complete hierarchy of extremal structures.

We approach this problem in the spirit of the Hungarian method. Given a Morse matching M^ϕ we want to compute an augmenting path p with maximum weight such that $M^\phi \Delta p$ is again a Morse matching. Loosely speaking, we compute a greedy approximation of (4.8). This approach is described in detail below.

For two dimensional manifold-like cell complexes this strategy is very efficient and easy to implement. In higher dimensions however, this approach encounters some problems and technical difficulties. We therefore restrict the presentation and evaluation to the two dimensional manifold-like case and refer to [GRP$^+$, Bau11] for details on the obstructions in higher dimensions.

4.5.1 Algorithm

The main problem for an algorithmic approach for the basic idea described above is the computation of the augmenting path p with maximum weight that results in another Morse matching. Fortunately, we can make use of Forman's cancellation theorem [For98b]. This theorem can be interpreted as a characterization of the set of augmenting paths that maintain the Morse property of a given matching. Using the graph theoretic formulation introduced in Section 4.1 this theorem can be stated as follows:

If two unmatched nodes are connected by exactly one p-separatrix s in a Morse matching $M \in \mathcal{M}^\phi$, then $M \triangle s$ is a Morse matching.

Note that there could possibly exist augmenting paths that maintain the Morse matching property which are not separatrices. However, we only consider the augmenting paths that are separatrices in our algorithm for two reasons. First, the computation of the separatrices is efficient and easy to implement, especially in the case of two dimensional manifold-like cell complexes. Secondly, the algorithm of the separatrix computation from Section 4.3 can be reused.

The pseudo code for our approximation algorithm is shown in Algorithm 8. The input consists of the cell Graph G and its edge weights ω. The output

consists of $V_{k_n}^{\phi}$ and a list of augmenting paths. Together, these can be used to reconstruct an arbitrary element of the sequence (4.8). The subfunction $getMaxUniqueSeparatrix(\ldots)$ returns the unique p-separatrix of maximum weight of the saddle u^1. The 2D manifold structure of the cell graph G implies that at most four p-streamlines emanate from u^1 and that these cannot split. The subfunction $getMaxUniqueSeparatrix(\ldots)$ therefore simply iterates all (up to four) p-separatrices that start in u^1. It then checks for uniqueness by comparing their end nodes and returns the unique p-separatrix with the largest weight. If there is no unique p-separatrix at all, then an empty path is returned with a weight of $-\infty$. Note that there are always two 0-streamlines emanating from u^1 and that these are always 0-separatrices. The 1-streamlines that emanate from u^1 however may end in the boundary of the manifold.

Algorithm 8 MorseMatchingSequence(G, ω)

Output: $AugPaths$, $V_{k_n}^{\phi}$

```
1:  M ← ∅, AugPaths ← ∅, heap ← ∅
2:  for all u¹ ∈ N do
3:      (path, weight) ← getMaxUniqueSeparatrix(G, ω, M, u¹)
4:      heap.push(u¹, weight)
5:  while heap ≠ ∅ do
6:      (u¹, weight) ← heap.pop()
7:      (path, weight) ← getMaxUniqueSeparatrix(G, ω, M, u¹)
8:      (nextNode, nextWeight) ← heap.top()
9:      if weight ≥ nextWeight then
10:         M ← M △ path
11:         if weight < 0 then
12:             AugPaths.push(path)
13:     else if −∞ < weight then
14:         heap.push(u¹, weight)
15: V_{k_n}^{φ} ← M
```

Line 1 initializes M as the empty matching, the list of augmenting paths $AugPaths$, and a priority queue $heap$. All nodes representing 1-cells are then inserted into this queue, ordered by the weight of their heaviest unique p-separatrix, in Lines 2-4. We then iterate over the queue (Line 5), remove the top element of the $heap$ (Line 6) and compute its heaviest unique p-separatrix (Line 7). This is necessary, as previous iterations may have affected this node. We now check whether this p-separatrix is the heaviest of all available unique p-separatrices. Assuming that augmenting the matching only decreases the weight returned by $getMaxUniqueSeparatrix(\ldots)$, it suffices to check whether the weight of u^1 is larger than the weight of the next element of the $heap$ (Lines 8-9). If this is the case, we augment the matching M by taking the symmetric difference of M and $path$ (Line 10) and store the augmenting

path if its weight is negative (Line 11-12). Otherwise, we reinsert u^1 into the *heap* with its new weight if it is larger than $-\infty$ (Line 13-14). When the *heap* is empty the algorithm terminates and returns an approximation of the heaviest maximum cardinality Morse matching $V^{\phi}_{k_n}$.

4.5.2 Evaluation

In this section, we present the resulting hierarchical extremal structure of Algorithm 8 and measure its performance on several data sets.

Robustness

To illustrate the robustness of our data analysis framework, we applied it to a synthetic data set depicted as a height field in Figure 4.10. The data set was produced by sampling the analytic function $f : [-1, 1]^2 \to \mathbb{R}$

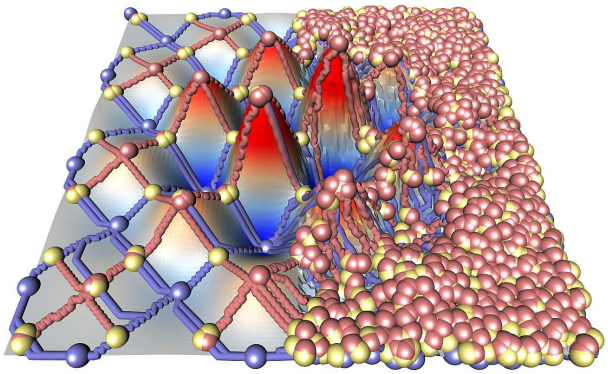
$$f(x, y) = \sin(10\,x)\,\sin(10\,y)\,e^{-3\,(x^2+y^2)} \tag{4.10}$$

on a uniform triangulation with $16k$ vertices. We then added uniform noise in the the range of $[-0.05, 0.05]$ to the sub domain $[0, 1] \times [-1, 1]$. We applied our algorithmic pipeline presented in Section 4.3 to this input data. The runtime for the computation of (4.8) using Algorithm 8 was less than a second on a standard workstation from 2008. Figure 4.10 shows the extremal structure of the initial combinatorial gradient field $V^{\phi}_{k_0}$, and two elements, $V^{\phi}_{k_n-23}$ and $V^{\phi}_{k_n-11}$, of the matching sequence (4.8). As can be seen in Figure 4.10a, $V^{\phi}_{k_0}$ includes the extremal structure induced by the noise. The simplified combinatorial gradient fields, however, only contain the dominant extremal structure present in f.

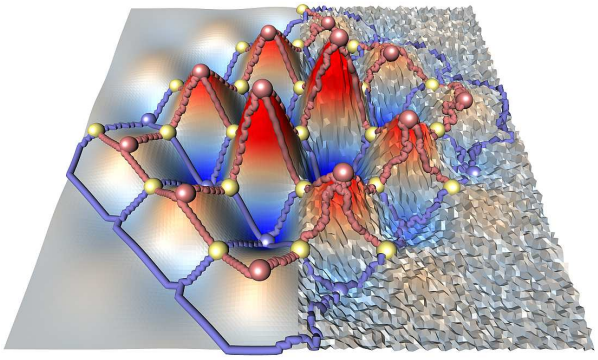
Application

Figure 4.11 illustrates the extraction of extremal lines in curvature fields of different surfaces. As described in [WG09], each point of a p-separatrix can be assigned an importance value, called separatrix persistence. The main idea of separatrix persistence is to measure the strength of monotony breaks with respect to the sequence of combinatorial gradient vector fields (4.8). For details how to incorporate this measure into our computational pipeline, we refer to [WG09].

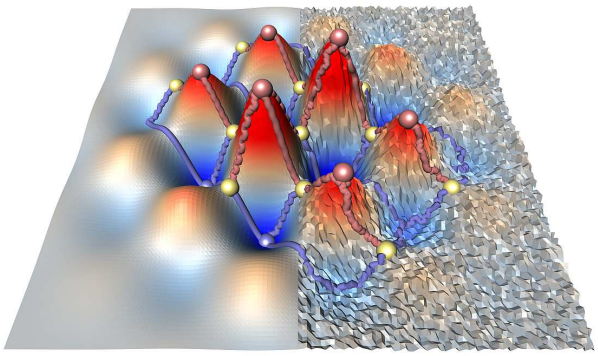
Separatrix persistence allows to discriminate spurious from dominant extremal lines. These lines are shown in Figure 4.11. Note that a reduction to the most dominant extremal parts destroys the connectivity of the extremal structure.



(a)



(b)



(c)

Figure 4.10 Synthetic noisy scalar field. Extremal structure of a) $V^{\phi}_{k_0}$, b) $V^{\phi}_{k_n-23}$ and c) $V^{\phi}_{k_n-11}$. Minima, saddles and maxima are depicted as blue, yellow and red spheres, while 0-separatrices and 1-separatrices are shown as blue and red lines.

Performance

All examples were computed on a workstation containing an Intel Core i7 860 CPU. The total running time for the computation of (4.8) using Algorithm 8 and the computation of separatrix persistence is shown in Table 4.2. The worst case complexity of Algorithm 8 is $O(n^3)$, where n denotes the number of edges in the triangulation. However, the empirical running time for practical applications is almost linear. The models are provided by Aim@Shape [Aim].

Surface Model	triangles	nodes in G	edges in G	time (sec)
screwdriver	54300	162902	325800	1
dinosaur	112384	337154	674304	2
knot	957408	2872224	5744448	24

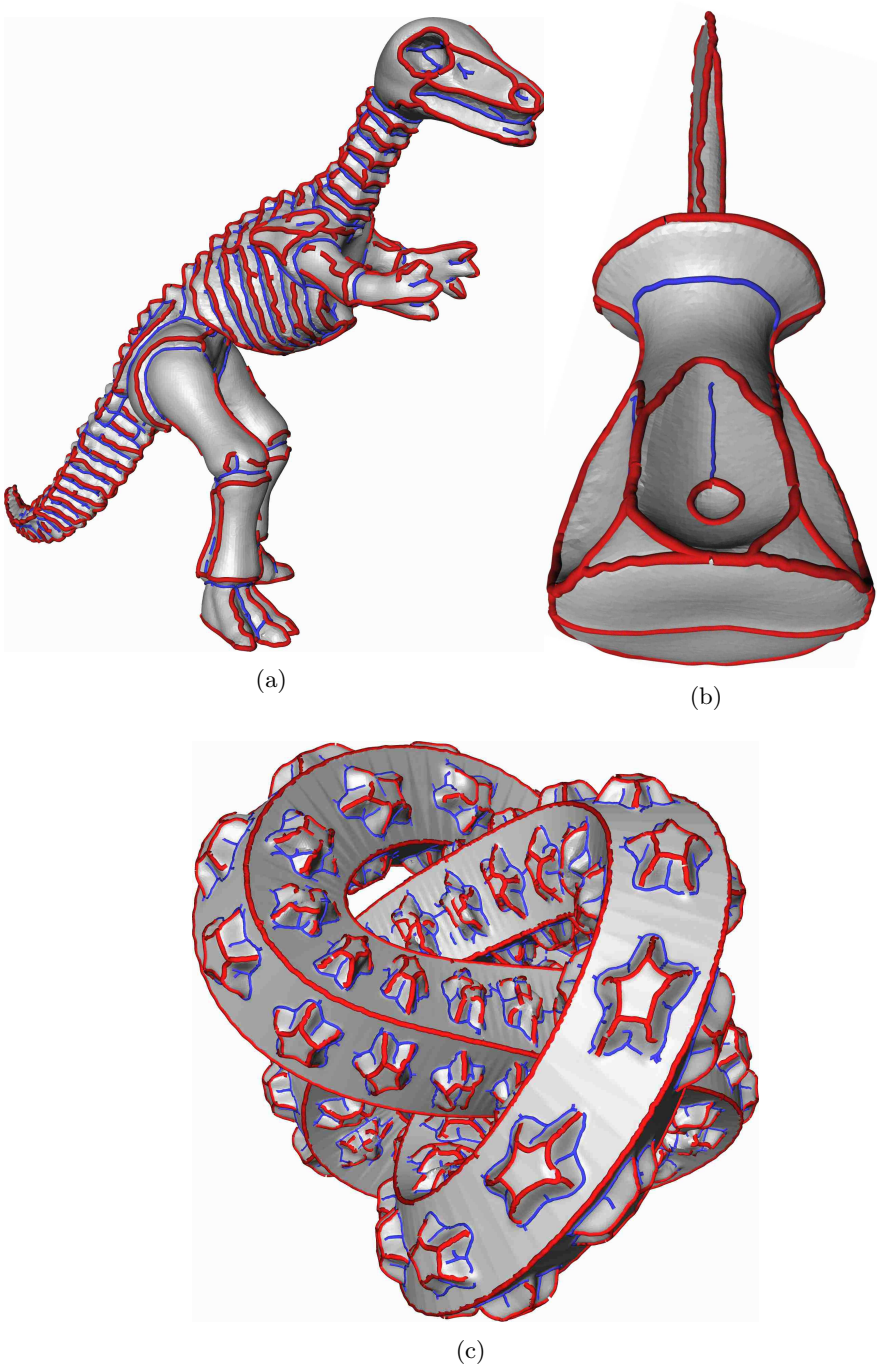
Table 4.2 Running time for Algorithm 8 with separatrix persistence [WG09] computation.

4.5.3 Discussion

The algorithm described in Section 4.5.1 is very simple, fast and delivers good results for practical data sets. However, its only mathematical guarantee is given by the provable consistence of the resulting extremal structure with the topology of the domain. While the resulting critical points seem to be fine, we have no guarantee of correctness.

To achieve such a correctness guarantee one can make use of the algorithm proposed by Robins et al. [RWS11] that computes an initial combinatorial gradient field with a (in a certain sense) provably correct set of critical points. Their algorithm works for (up to) three dimensional manifold-like cell complexes.

It was recently shown [DLL$^+$10] that our hierarchy of extremal structures is strongly related to persistent homology if we use the algorithm by Robins et al. for an initial combinatorial gradient. Note that this result only holds for two dimensional manifold-like cell complexes – in higher dimensions a counter example has been published [Bau11].



(a) (b) (c)

Figure 4.11 Extremal lines in curvature fields. For all surface models, the first and second principal curvatures κ_1 and κ_2 are computed. a), b) and c) show the most dominant parts of 0-separatrices (blue) in κ_1 and 1-separatrices (red) in κ_2. Images included with permission of the authors of [WG09]

4.6 Divergence-free vector fields

This section introduces a robust and provably consistent algorithm for the topological analysis of divergence-free 2D vector fields. Note that divergence-free vector fields cannot just be treated as general vector fields using the algorithm presented in Section 4.4.1: center-like critical points are stable features of divergence-free vector fields while they are unstable in the set of general vector fields.

In the next section we will show that the case of two dimensional divergence-free vector fields can in fact be treated using the algorithm presented in Section 4.5.1. Note that in contrast to the preceding sections, the two dimensional assumption is crucial in this case.

4.6.1 Theory

This section shows how theorems from classical Morse theory can be applied in the context of 2D divergence-free vector fields. For completeness, we include the general theorems of Morse theory presented in Section 3.3 for the two dimensional case.

Extremal structure of divergence-free vector fields

A 2D vector field v is called divergence-free if $\nabla \cdot v = 0$. This class of vector fields often arises in practice, especially in the context of computational fluid dynamics. For example, the vector field describing the flow of an incompressible fluid, like water, is must be divergence-free. The points at which a vector field v is zero are called the critical points of v. They can be classified by an eigenanalysis of the Jacobian Dv at the respective critical point. In the case of divergence-free 2D vector fields one usually distinguishes two cases [HH89]. If both eigenvalues are real, then the critical point is called a saddle. If both eigenvalues are imaginary, then the critical point is called a center. Note that one can classify a center furthermore into clockwise rotating (CW-center) or counter-clockwise rotating (CCW-center) by considering the Jacobian as a rotation.

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

Morse theory in two dimensions

The critical points of a vector field are often called topological features. One justification for this point of view is given by Morse theory [Mil63]. Loosely

speaking, Morse theory relates the set of critical points of a vector field to the topology of the domain. For example, it can be proven that every continuous vector field on a sphere contains at least one critical point.

To make things more precise we restrict ourselves to gradient vector fields defined on a closed oriented surface. The ideas presented below work in principal also for surfaces with boundary, but the notation becomes more cumbersome. To keep things simple, we therefore assume that the surface is closed. We further assume that all critical points are first order, i.e. the Jacobian has full rank at each critical point. Let c_0 denote the number of minima, c_1 the number of saddles, c_2 the number of maxima, and g the genus of the surface. We then have the Poincaré-Hopf theorem

$$c_2 - c_1 + c_0 = 2 - 2g, \tag{4.11}$$

the weak Morse inequalities

$$c_0 \geq 1, \qquad c_1 \geq 2g, \qquad c_2 \geq 1, \tag{4.12}$$

and the strong Morse inequality

$$c_1 - c_0 \geq 2g - 1. \tag{4.13}$$

Helmholtz-Hodge decomposition

To apply these theorems from Morse theory to a divergence-free vector field v we can make use of the Helmholtz-Hodge decomposition [Hel58]. Let $\nabla \times \psi = (\partial_y \psi, -\partial_x \psi)$ denote the curl operator in 2D. We then have the orthogonal decomposition

$$v = \nabla\phi + \nabla \times \psi + h. \tag{4.14}$$

We can thereby uniquely decompose v into an irrotational part $\nabla\phi$, a solenoidal part $\nabla \times \psi$, and a harmonic part h, i.e. $\Delta h = 0$. Due to the assumption that the surface is closed, the space of harmonic vector fields coincides with the space of vector fields with zero divergence and zero curl [Sho09]. Since v is assumed to be divergence-free we have $0 = \nabla \cdot v = \nabla \cdot \nabla\phi$ which implies $\phi = 0$ due to (4.14). The harmonic-free part $\hat{v} = v - h$ can therefore be expressed as the curl of a scalar valued function

$$\hat{v} = \nabla \times \psi. \tag{4.15}$$

Stream function

The function ψ is usually referred to as the stream function [Pan84]. Let $\hat{v}^\perp = (v_2, -v_1)$ denote the point-wise perpendicular vector field of $\hat{v} = (v_1, v_2)$. The gradient of the stream function is then given by

$$\nabla\psi = \hat{v}^\perp. \tag{4.16}$$

Note that $\hat{v}$ has the same set of critical points as $\hat{v}^{\perp}$. The type of its critical points is however changed: CW-center become minima, and CCW-center become maxima. Since (4.16) shows that $\hat{v}^{\perp}$ is a gradient vector field, we can use this identification to see how (4.11)-(4.13) can be applied to the harmonic-free part of divergence-free 2D vector fields.

Implications

The dimension of the space of harmonic vector fields is given by $2g$ [Sho09]. A vector field defined on a surface which is homeomorphic to a sphere is therefore always harmonic-free, i.e. $\hat{v} = v$. Every divergence-free vector field on a sphere which only contains first order critical points therefore satisfies (4.11)-(4.13). For example, every such vector field contains at least one CW-center and one CCW-center.

Due to the practical relevance in Section 4.6.3 we note that every divergence-free vector field defined on a contractible surface can be written as the curl of a stream function ψ as shown by the Poincaré-Lemma. For such cases, the point-wise perpendicular vector field can therefore also be directly interpreted as the gradient of the stream function.

4.6.2 Algorithm

We now describe how we can apply computational discrete Morse theory to divergence-free vector fields. Let v denote a divergence-free vector field defined on an oriented surface S. The first step is to compute the harmonic-free part $\hat{v}$ of v. If S is contractible or homeomorphic to a sphere, then v is itself the curl of a stream function ψ, i.e. $\hat{v} = v$. Otherwise, we need to compute the Helmholtz-Hodge decomposition (4.14) of v to get its harmonic part. To do this, one can employ the algorithms described in [PPL^{+}10, PP02, TLHD03].

We now make use of the fact that the point-wise perpendicular vector field $\hat{v}^{\perp}$ has the same critical points as $\hat{v}$. Due to (4.15), we know that $\hat{v}^{\perp}$ is a gradient vector field. To compute and classify the critical points of the divergence-free vector field $\hat{v}$ it therefore suffices to analyze the gradient vector field $\hat{v}^{\perp}$.

One approach to analyze the gradient vector field $\hat{v}^{\perp}$ would be to compute a scalar valued function ψ such that $\hat{v}^{\perp} = \nabla\psi$. One can then apply one of the algorithms mentioned in Section 2.2 to extract a consistent set of critical points. In this section, we instead apply the algorithm proposed in Section 4.5.1 to directly analyze the gradient vector field $\hat{v}^{\perp}$ by computing (4.7) for $\hat{v}^{\perp}$ in the context of combinatorial gradient fields.

The main benefit of this approach is that it allows us to consider $\hat{v}^{\perp}$ as a gradient vector field even if it contains a small amount of curl. This is a common problem in practice, since a numerical approximation or measurement

of a divergence-free field often contains a small amount of divergence. By adapting the general approach presented in this chapter, we can directly deal with such fields with no extra pre-processing steps.

Note that the importance measure for the critical points of a gradient vector field has a nice physical interpretation in the case of rotated stream functions. The importance measure is defined by the height difference of a certain pairing of critical points. The height difference between two points of the stream function is the same as the amount of flow passing through any line connecting the two points [Pan84]. This allows us to differentiate between spurious and structurally important critical points in divergence-free 2D vector fields, as will be demonstrated in the next section.

4.6.3 Evaluation

The purpose of this section is to evaluate the practical properties of our algorithm and determine the physical relevance of the hierarchy of extremal structures.

Robustness

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

$$v(x, y) = \nabla \times \left(\sin(6\,x)\, \sin(6\,y)\, e^{-3\,(x^2+y^2)} \right) \tag{4.17}$$

on the domain $[-1, 1]^2$ with a uniform 512^2 grid. A LIC image [SH95] of this divergence-free vector field is shown in Figure 4.12, left. To simulate a noisy measurement of this vector field, we added uniform noise with a range of $[-1, 1]$ to this data set. A LIC image of the resulting quasi-divergence-free vector field is shown in Figure 4.12, right. Since the square is a contractible domain, we can directly apply the algorithm described in Section 4.6.2 to both data sets and extracted the 23 most important critical points. As can be seen in Figure 4.12, our method is able to effectively deal with the noisy data.

Application

To illustrate the physical relevance of the importance measure for the extracted critical points we consider a model example from computational fluid dynamics [NSA+08]. Figure 4.13, top, shows a LIC image of a simulation of the flow behind a circular cylinder – the cylinder is on the left of the shown data set. Since we are considering only a contractible subset of the data set, we can directly apply the algorithm described in Section 4.6.2. Note that due to a uniform sampling of this data set a small amount of divergence was introduced. The divergence is depicted in Figure 4.13, bottom. The data set exhibits the well-known Kármán vortex street [Pan84] of alternating clockwise

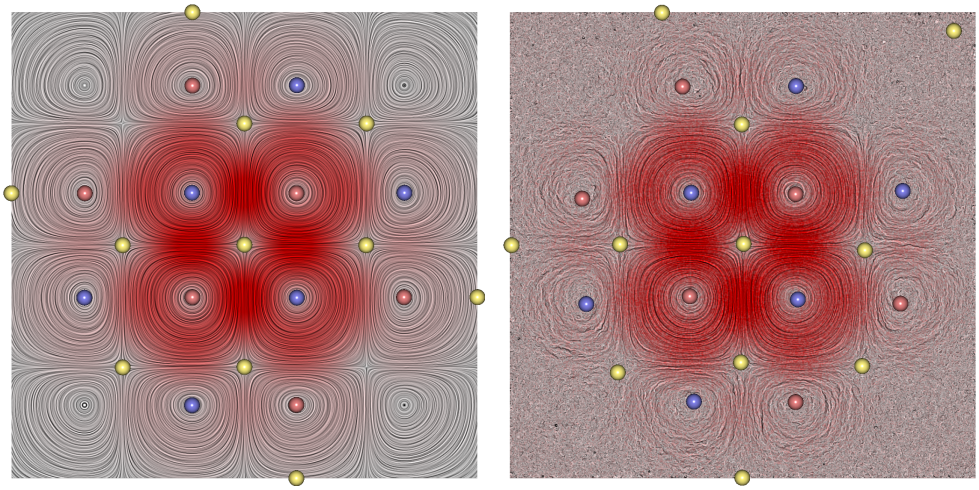


Figure 4.12 A synthetic divergence-free vector field is depicted using a LIC image colored by magnitude (red = high). The critical points of $V^{\phi}_{k_n-11}$ are shown. The saddles, CW-centers, and CCW-centers are depicted as yellow, blue, and red spheres. Left: the original smooth vector field. Right: a noisy measurement of the field depicted on the left.

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

Performance

The running time of our algorithm is 47 seconds for a surface with one million vertices using an Intel Core i7 860 CPU with 8GB RAM.

4.6.4 Discussion

We presented an algorithm for the extraction of critical points in 2D divergence-free vector fields. In contrast to previous work this algorithm is provably consistent in the sense of Morse theory for divergence-free vector fields as presented in Section 4.6.1. It also allows for a consistent simplification of the set of critical points which enables the analysis of noisy data as illustrated in Figure 4.12. The computed importance measure has a physical relevance as shown in Figure 4.13, and allows to discriminate between dominant and spurious critical points in a data set. By combinatorially enforcing the gradient vector field property we are able to directly deal with data sets with only near zero divergence (see Figure 4.13, bottom).

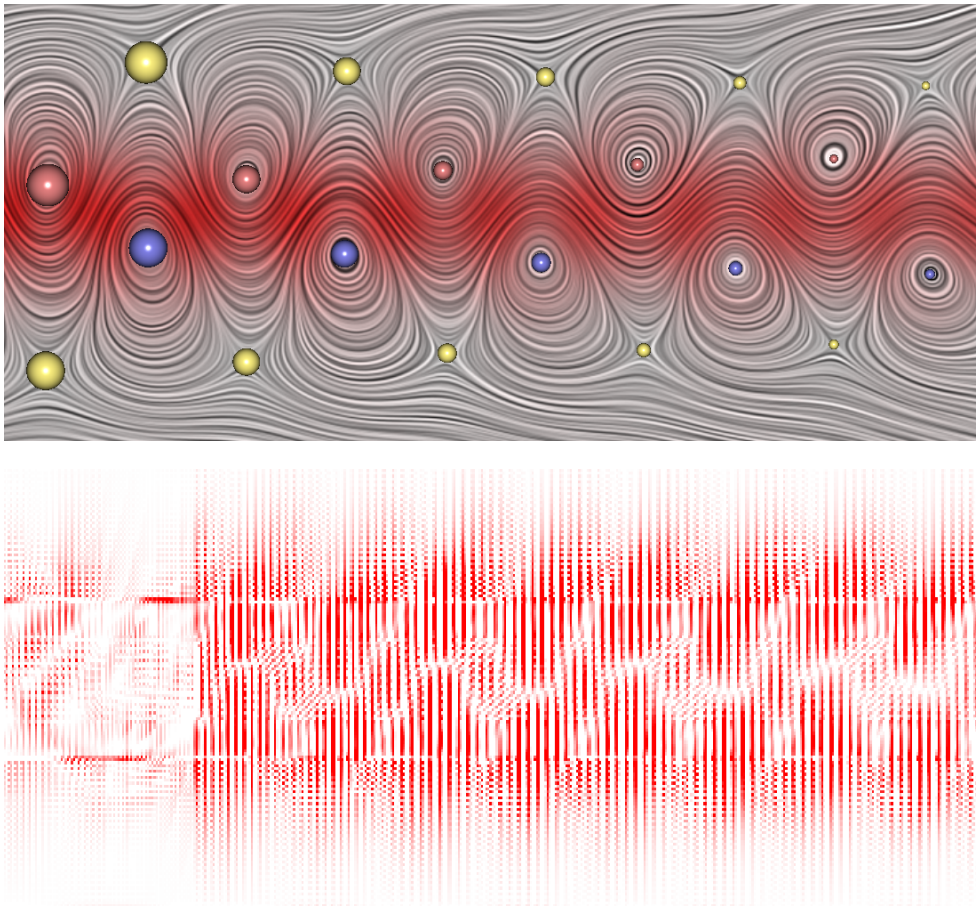


Figure 4.13 Top: A quasi-divergence-free vector field of the flow behind a circular cylinder is depicted using a LIC image colored by magnitude. The saddles, CW-centers, and CCW-centers are depicted as yellow, blue, and red spheres and are scaled by our importance measure. Bottom: the divergence of the data set is shown using a colormap (white: zero divergence, red: high divergence).

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

4.7 Time-dependent scalar fields

Time-dependent scalar data arises in many scientific disciplines. To analyze such data, the extraction of minima, saddles, and maxima of each individual time step has been proven useful. These point features of the data are often called critical points. To understand the dynamic behavior of time-dependent data, it can be beneficial to analyze the temporal evolution of these critical points.

To enable an efficient quantification of the temporal evolution of the critical points, we can track them over time. In this section, we call such a tracked critical point a critical line of the data. Many different algorithms that extract critical lines have been proposed, see Section 2.3 for an overview.

For smooth functions, the Feature Flow Field method [TS03] provides a particularly sound mathematical foundation. Given a smooth time-dependent scalar field, the critical lines are therein implicitly defined by streamlines in a higher dimensional derived vector field.

While this method works well for smooth functions, its application to functions that are only continuous is problematic as derivatives have to be computed. To circumvent this problem, derivative free algorithms employing concepts from algebraic topology have been developed recently, see Section 2.3.

The main remaining weakness of the available algorithms is their inability to handle noisy data in a meaningful way. Such data usually contains an overwhelming number of critical lines that hinder meaningful visual data analysis. To reduce the number of critical lines, one typically smoothes the data or discards short critical lines. Both approaches can be problematic. A simple smoothing of the data may remove important critical lines and affect the spatial position of the critical lines, see Figure 4.18 for an example. Discarding short critical lines may remove an important and stable, but short lived feature. See Figure 4.21 for an example of such a short but important critical line.

This section proposes a combinatorial algorithm that is able to track critical points in noisy data. This robustness is achieved by combining Forman's notion of a combinatorial gradient field [For98a] with the notion of persistence proposed by Edelsbrunner et al. [ELZ02]. Persistence is a well founded importance measure for critical points. Together, these concepts enable a robust and consistent combinatorial representation of the gradient of a scalar function.

A definition for a critical line of a sequence of combinatorial gradient fields was recently proposed by King et al. [KKM08]. The basic idea is similar to the continuous Feature Flow Field method - a higher dimensional field is constructed in which the critical lines are given by combinatorial streamlines. We therefore refer to the higher dimensional field as a Combinatorial Feature Flow Field in this section. We formalize this concept of critical lines in com-

binatorial gradient fields using a graph theoretic formulation in Section 4.7.1.

Our main contribution is the introduction of the first efficient algorithm that extracts the critical minima, saddle, and maxima lines in two dimensional time-dependent scalar fields using Combinatorial Feature Flow Fields.

The efficiency of our algorithm is achieved by pointing out that, at least in 2D, the combinatorial critical lines as defined in [KKM08] can in fact be computed without considering the higher dimensional combinatorial feature flow field. We provide an informal proof of the equivalence of the original definition of combinatorial critical lines to our simplified definition in Section 4.7.2.

The proposed algorithm has many valuable properties. It has a reasonable running time (see Table 4.3) and is naturally formulated in an out-of-core fashion enabling the analysis of large data sets as only two subsequent time steps have to be kept in memory. The input consists of a regular cell complex, so the algorithm can deal with many widely used representations of discrete data like triangulations, quadrangulations, or a mixture of these. It contains only one easily-tuned computational parameter, the persistence threshold σ, used to construct the combinatorial representation of the gradient fields.

Due to the combinatorial nature of our algorithm, we can formulate a natural spatio-temporal importance measure for the resulting critical lines called Integrated Persistence (see Section 4.7.1).

4.7.1 Theory

The purpose of this section is to introduce the reader to the main concepts that build the mathematical foundation for our combinatorial tracking algorithm described in detail in Section 4.7.2. We will also propose a time-aware importance measure for the tracked critical points that is based on the notion of persistence below.

To define the notion of a combinatorial feature flow field (CFFF) we rely on the concept of a combinatorial gradient field (CGF) introduced in Section 3.3.3 in the graph theoretic formulation described in Section 4.1.

When we deal with noisy data, the corresponding CGF contains a huge number of minima, saddles, and maxima. Fortunately, the theoretical foundation of CGF [For98b] allows for a consistent removal of these spurious features. Suppose there is a unique separatrix connecting a saddle to a maximum or minimum. Reversing this separatrix results in a CGF without this pair of critical points, see [For01] for an explanation of this property. When we simplify a CGF using this idea we have to decide on the order of the simplifications. A well founded order is given by persistence [EH08].

Note that the need for a simplified CGF implies that one cannot make direct use of the algorithm proposed in [Gyu08] since it computes a simplified Morse-Smale complex and not a simplified CGF. Reconstructing a CGF from a

Morse-Smale complex is a nontrivial problem, especially in higher dimensions.

To track critical points in noisy data, we compute a CGF with a given persistence threshold σ. For example, if the noise is in the range $[-\epsilon, \epsilon]$, then it suffices to compute its CGF with a persistence threshold of $\sigma = 2\epsilon$ to remove all noise induced critical points. This means that the resulting CGF does not contain any pair of uniquely connected critical points whose difference of scalar values is smaller than σ. For more information on the connection between discrete Morse theory and persistence simplification we refer the interested reader to [Bau11].

Combinatorial Feature Flow Fields

Using the combinatorial representation of the gradient fields defined above, we will now describe the combinatorial feature flow field concept introduced in [KKM08] that allows us to track critical points in our graph theoretical framework. This formulation enables an efficient and simple implementation described in Section 4.7.2.

Given a sequence of combinatorial gradient fields $(V_t)_{t=0,1,2,\ldots,T}$ on a *fixed* cell complex C of a 2D manifold we now define the notion of a combinatorial feature flow field (CFFF) that allows us to track the critical points in (V_t). For simplicity, we assume $T = 1$ as the general case follows easily. We first construct the cell graph of $C \times [0, 1]$ using the graph theoretic formulation introduced in Section 4.1.

For a depiction of a simple example of the rather technical construction that follows, we refer to Figure 4.14. We start the construction of $G_{C\times[0,1]}$ with three copies G_C^1, G_C^2, G_C^3 of the cell graph G_C. We then add edges to this graph that connect the corresponding nodes of G_C^1 with G_C^2 and G_C^2 with G_C^3. The label p of each node in G_C^2 is then increased by one. For example, if u^p is a node of the second copy that corresponds to the node w^2 of the first copy, then $p = 3$.

We can now define the forward tracking field $V_{[0,1]}$, a CGF of $G_{C\times[0,1]}$. We first use the matching V_0 to define a matching in G_C^1 and G_C^2 (see Figure 4.14, middle). For G_C^3 we use the matching V_1. We then add all edges to the matching of $G_{C\times[0,1]}$ that connect a critical point of V_0 with a node of G_C^2.

Constructing a forward tracking field $V_{[0,T]}$ for the whole sequence of combinatorial gradient fields (V_t) can be done iteratively: if we have a forward tracking field for $V_{[0,k]}$, we get $V_{[0,k+1]}$ as the union of $V_{[0,k]}$ and $V_{[k,k+1]}$. The backward tracking field $V_{[T,0]}$ can be defined by reversing the order of the sequence (V_t). As proven in [KKM08], the forward tracking field defined above is indeed a combinatorial gradient field as it does not contain any periodic orbits. Also, the only critical cells of this CGF are the cells that are critical in V_T.

We are now in a position to give a precise definition of the space-time

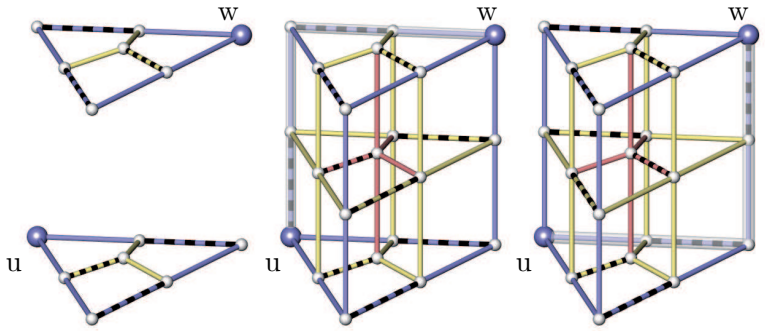


Figure 4.14 Combinatorial feature flow fields - basic definitions. Left: Two subsequent combinatorial gradient fields V_0 and V_1. Middle: Forward tracking field $V_{[0,1]}$. Right: Backward tracking field $V_{[1,0]}$. The minima (blue) u in V_0 and w in V_1 are tracked as there is a combinatorial 0-streamline (transparent) in $V_{[0,1]}$ and a combinatorial 0-streamline in $V_{[1,0]}$ that connect u and w.

relation of critical points in this combinatorial setting. Let u^p and w^p denote critical points in (V_t). We say u^p and w^p are connected if and only if there is a combinatorial p-streamline connecting u^p with w^p within $V_{[0,T]}$ and a combinatorial p-streamline connecting u^p with w^p within $V_{[T,0]}$. For future reference, we call the set of lines that connect the critical points of (V_t) the critical lines of (V_t). Note that in principal this definition allows for splitting and merging critical saddle lines. While our implementation allows for this behavior we have not observed any such critical saddle lines in our numerical experiments.

The presented approach is related to the continuous Feature Flow Field method [TS03] - both approaches for the tracking of critical points define a higher dimensional field where the critical points can be tracked by streamlines. We therefore refer to the approach presented in this Section as the Combinatorial Feature Flow Field method (CFFF).

Integrated Persistence

We propose an importance measure for the critical lines of a sequence of T scalar fields (f_t) defined on a 2D manifold. To incorporate the spatial importance of the critical points that make up the critical line we can make use of the notion of persistence [ELZ02]. Loosely speaking, persistence measures the stability of the critical points with respect to perturbations of the data values. We now define an importance measure for a critical line L as the sum of the persistence values of the critical points that make up the line divided by the total number of time steps in the data set T. For future reference we refer to this measure as Integrated Persistence.

Note that in some sense Integrated Persistence is a spatio-temporal importance measure. A short, but spatially persistent critical line, is considered as important as a long critical line with low spatial persistence. Figures 4.19 and 4.21 demonstrate the physical relevance of Integrated Persistence.

4.7.2 Algorithm

In this section, we describe our combinatorial tracking algorithm in detail. We will first give an overview of the algorithmic pipeline, describing the input, output and out-of-core approach. We then describe how we can efficiently track critical points. This section is concluded with a detailed description of our algorithm including pseudo-code to ensure a good reproducibility of the results presented in Section 4.7.3.

Overview

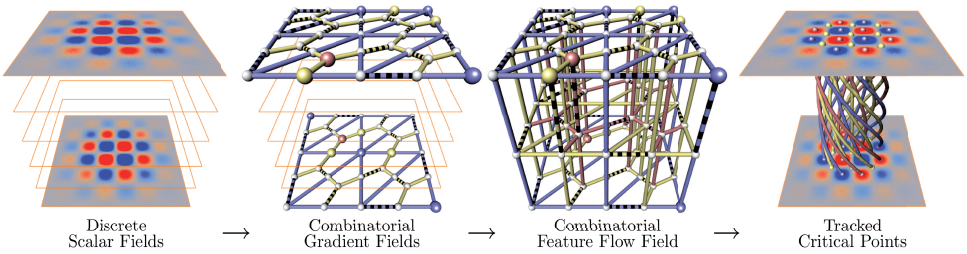


Figure 4.15 Computational pipeline of the algorithm described in Section 4.7.2.

The input of our algorithm consists of a regular cell complex C of a 2D manifold and a sequence of scalar fields (f_t) defined on the 0-cells of C. A simple example of such input data is a triangulation or a quadrangular mesh with a sequence of scalar values defined on each vertex. We then compute a sequence of combinatorial gradient fields (V_t) with persistence threshold σ that represents the gradient of the input data in a discrete fashion. To deduce an importance measure for our result we will also require the persistence values of the critical points contained in (V_t).

A closer inspection of the definition given in Section 4.7.1 reveals that we can compute all critical lines contained in (V_t) in a streaming fashion - it is sufficient to compute the critical lines of each consecutive pair of the sequence (V_t). Due to the combinatorial nature of the critical lines they can easily be merged afterwards to get the result for the complete data set. The importance measure for a critical line introduced in Section 4.7.1 can be computed by adding the persistence values of the critical points contained in the line. See Figure 4.15 for an overview of the overall algorithm.

Efficient Extraction of Critical Lines in CFFF

As described above, it suffices to track the critical points for each consecutive pair (V_k, V_{k+1}) of the sequence of combinatorial gradient fields (V_t). As defined in Section 4.7.1, a critical point of V_k is connected to a critical point of V_{k+1} if and only if there is a combinatorial streamline in the forward tracking field $V_{[k,k+1]}$ and a combinatorial streamline in the backward tracking field $V_{[k+1,k]}$ connecting the two points.

The goal is now to define a simple algorithm that finds all pairs of critical points that satisfy this condition. It will be shown that we actually do not need to construct the higher dimensional cell graph $G_{C\times[0,1]}$. This significantly reduces the runtime, memory consumption, and greatly simplifies the implementation of our algorithm.

For a depiction of the following argument, we refer to Figure 4.14. We start with the minima. Let u denote a minimum in V_k. When we iterate the combinatorial 0-streamlines of the forward tracking field that start in u we see that there is only a single streamline that ends in a minimum of V_{k+1}. This is due to two reasons. First, the structure of the forward tracking field implies that the only way to reach V_{k+1} is to start with the matched edge adjacent to u. Second, a combinatorial streamline whose first node is not a 1-cell and whose first edge is matched, is uniquely defined as it cannot split. For an explanation of this basic property of combinatorial streamlines we refer the reader to [For01]. The same arguments can be employed to show that there is only a single streamline connecting a minimum of V_{k+1} to a minimum of V_k in the backward tracking field.

Tracking minima is therefore a rather simple procedure. Given a minimum u in V_k we find its only possible partner w in V_{k+1} by computing the unique streamline in the forward tracking field that starts in u with a matched edge. We then compute the unique streamline in the backward tracking field that starts in w with a matched edge. If this streamline ends in u, then u and w are connected in the sense of the definition given in Section 4.7.1.

Note that we do not actually need to construct the forward and backward tracking fields to compute these combinatorial streamlines. It suffices to trace them in the given pair of CGFs V_k and V_{k+1} as can be seen in Figure 4.14.

The maxima can be tracked in the same way, we only have to switch forward and backward tracking fields: the maxima of V_{k+1} have only a single partner in V_k in the forward tracking field, and the maxima of V_k have only a single partner in V_{k+1} in the backward tracking field.

While tracking minima and maxima has been proven to be rather simple, tracking of saddles seems to be a very daunting task as the combinatorial 1-streamlines in the higher dimensional tracking fields may merge and split (see Figure 4.16). On first sight, it seems that the only way to compute the critical saddle lines is a brute-force depth-first-search in the tracking fields. However,

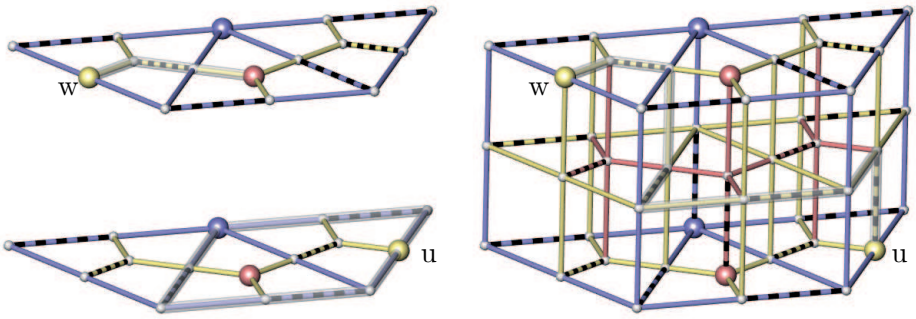


Figure 4.16 Saddle tracking in CFFF. Left: Two subsequent combinatorial gradient fields V_0 and V_1 on three triangles. Right: Forward tracking field $V_{[0,1]}$. The saddles (yellow) u in V_0 and w in V_1 are connected in $V_{[0,1]}$ by a combinatorial 1-streamline (transparent) that connects the corresponding nodes. Note that the minima lines (transparent) of the saddle of V_0 (bottom-left) intersect the maxima lines (transparent) of the saddle in V_1 (top-left).

a close inspection of the structure of the tracking fields reveals that this is not actually necessary.

Consider the 1-streamlines of the forward tracking field that start in a saddle u of V_k and end in a saddle w of V_{k+1}. If we think of the graph of the forward tracking field as consisting of three layers (the three copies of G_C), we can observe three properties of these streamlines (see Figure 4.16 for an example):

1. The layer of the nodes of the streamlines only increases and the only node of the bottom layer is the node in which we start.

2. The section of these streamlines that runs through the second layer follows the 0-streamlines of V_k that start in u.

3. The section of these streamlines that runs through the third layer follows the 1-streamlines of V_{k+1} and ends in w.

These properties imply that there is a combinatorial 1-streamline in the forward tracking field that connects u with w if and only if the 0-streamlines of V_k that start in u intersect the 1-streamlines of V_{k+1} that end in w. Similarly, there is a combinatorial 1-streamline in the backward tracking field that connects w with u if and only if the 0-streamlines of V_{k+1} that start in w intersect the 1-streamlines of V_k that end in u.

Instead of a brute-force search in the higher dimensional cell graph $G_{C\times[0,1]}$, it therefore suffices to intersect the separatrices of u defined by V_k with the separatrices of w defined by V_{k+1} in the low dimensional cell graph G_C. This simplifies the following tracking algorithm significantly.

Algorithm 9 Main CFFF algorithm

Input: C, (f_t), T, σ
Output: All critical lines in $V_{[0,T]}$

1: $G_C \leftarrow$ constructCellGraph(C)
2: **for** $k = 0$ to $T - 1$ **do**
3: $V_k \leftarrow \text{CGF}(G_C, f_k, \sigma)$
4: $V_{k+1} \leftarrow \text{CGF}(G_C, f_{k+1}, \sigma)$
5: lines $\leftarrow$ lines $\cup$ trackMinMax(G_C, V_k, V_{k+1})
6: lines $\leftarrow$ lines $\cup$ trackSaddles(G_C, V_k, V_{k+1})

Implementation

The main algorithm that tracks the critical points of a sequence of discrete scalar fields (f_t) defined on the 0-cells of a cell complex C is given in Algorithm 9. Line 1 constructs the cell graph G_C of the cell complex C as defined in Section 4.1. The CGF subfunction called in Lines 3 and 4 computes a combinatorial gradient field with a persistence threshold σ. To do this we first compute an initial CGF using the algorithm proposed in [RWS11].

It is proven that the resulting CGF contains exactly the critical points contained in the discrete input data. We can thereby state an estimate for the precision of our method. Suppose that we sample an analytic function with a uniform grid whose cells have size h. Furthermore, assume that h is sufficiently small so that all critical points of the analytic function are represented in the discrete representation. Then the distance between the exact critical points and the critical points in the CGF is smaller than h.

To compute the simplified CGF we follow the approach presented in Section 4.3. We thereby compute the whole sequence of simplified CGFs, which has the advantage of allowing the user to quickly select the appropriate simplification threshold σ in a post processing step. For the persistence values, we employ the matrix reduction algorithm presented in [CSEM06].

Line 5 and 6 extract the critical minima, maxima and saddle lines of the current pair of CGFs as defined in Section 4.7.1 using the simplified but equivalent definition presented in Section 4.7.2.

To compute the critical lines we need to compute a lot of combinatorial p-streamlines in a given CGF V_k. The pseudo-code for such a combinatorial streamline integrator is given in Algorithm 10. Almost all computational time of the main Algorithm 9 is spent integrating such lines which makes the performance of this algorithm crucial for the overall runtime. Note that due to the structure of the cell graph G_C and the matching property of V_k, there cannot exist multiple edges that fulfill the condition in Line 3. Of course, an actual implementation would not take the complement of the matching in each iteration (Line 5). One would rather simply switch the if condition in

Algorithm 10 Combinatorial Streamline Integrator: *traceLine(...)*

Input: $G_C = (S, L)$, $V_k \subset L$, $u \in S$, $p = 0, 1$
Output: Combinatorial p-streamline that starts in u

```
1: loop
2:    Line.append( u )
3:    if there exists w: {u, w} = ℓ^p ∈ V_k then
4:       u ← w
5:       V_k ← V_k^c
6:    else
7:       return
```

Algorithm 11 Min and max tracking algorithm: *trackMinMax(...)*

Input: $G_C = (S, L)$, $V_k \subset L$, $V_{k+1} \subset L$
Output: All critical min/max lines in $V_{(k,k+1)}$

```
1: for all u^p ∉ S(V_k) and p ≠ 1 do
2:    p ← max(0, p − 1)
3:    Line ← traceLine(G_C, V_{k+1}, u, p)
4:    w ← Line.last()
5:    Line ← traceLine(G_C, V_k, w, p)
6:    if Line.last() = u then
7:       MinMaxCritLines.append( {u, w} )
```

Line 3.

Using Algorithm 10, we can compute the critical minima and maxima lines as shown in Algorithm 11. For each minimum or maximum u (Line 1) of the first CGF V_k, we integrate the corresponding combinatorial p-streamline in V_{k+1} (Line 3). We now take the last point w of this streamline as the start point of a streamline in V_k (Lines 4, 5). If this streamline comes back to u (Line 6), we found a critical line in $V_{(k,k+1)}$ and append the pair $\{u, w\}$ to the result.

To compute the critical saddle lines, we need to compute the separatrices of the saddles. A simple method that returns the separatrices of type p of the saddle u in the CGF V_k is given in Algorithm 12. We iterate over all adjacent edges ℓ of the saddle u of the given type p (Line 1) and integrate the combinatorial p-streamline that starts in the end point w of ℓ (Line 2). This line is then appended to the separatrices (Line 3).

Using Algorithm 12, we can now trace the critical saddle lines as shown in Algorithm 13. Each saddle is appended to the nodes covered by its separatrices of type 0 (Lines 1-8). We then iterate over each saddle u of V_k (Line 9). The possible saddle partners for u in V_{k+1} are then given as the union of the saddles in V_{k+1} whose separatrices of type 0 are intersected by the separatrices of type 1 of u (Lines 10-13). For each such partner w we then iterate its partners in

Algorithm 12 Separatrix Integrator: *traceSeps(...)*

Input: $G_C = (S, L)$, $V_k \subset L$, $u \in S$, $p = 0, 1$
Output: All combinatorial p-streamlines that start in saddle u

```
1: for all {u, w} = ℓ^p ∈ L do
2:     Line ← traceLine(G_C, V_k, w, p)
3:     Separatrices.append( Line )
```

Algorithm 13 Saddle tracking algorithm: *trackSaddles(...)*

Input: $G_C = (S, L)$, $V_k \subset L$, $V_{k+1} \subset L$
Output: All critical saddle lines in $V_{(k,k+1)}$

```
1: for all u^1 ∉ S(V_k) do
2:     minLines_u ← traceSeps(G_C, V_k, u, 0)
3:     for all w ∈ minLines_u do
4:         saddles_k[w].append(u)
5: for all u^1 ∉ S(V_{k+1}) do
6:     minLines_u ← traceSeps(G_C, V_{k+1}, u, 0)
7:     for all w ∈ minLines_u do
8:         saddles_{k+1}[w].append(u)
9: for all u^1 ∉ S(V_k) do
10:    maxLines_u ← traceSeps(G_C, V_k, u, 1)
11:    partners_u ← ∅
12:    for all w ∈ maxLines_u do
13:        partners_u ← partners_u ∪ saddles_{k+1}[w]
14:    for all w ∈ partners_u do
15:        maxLines_w ← traceSeps(G_C, V_{k+1}, w, 1)
16:        partners_w ← ∅
17:        for all v ∈ maxLines_w do
18:            partners_w ← partners_w ∪ saddles_k[v]
19:        if u ∈ partners_w then
20:            SaddleCritLines.append( {u, w} )
```

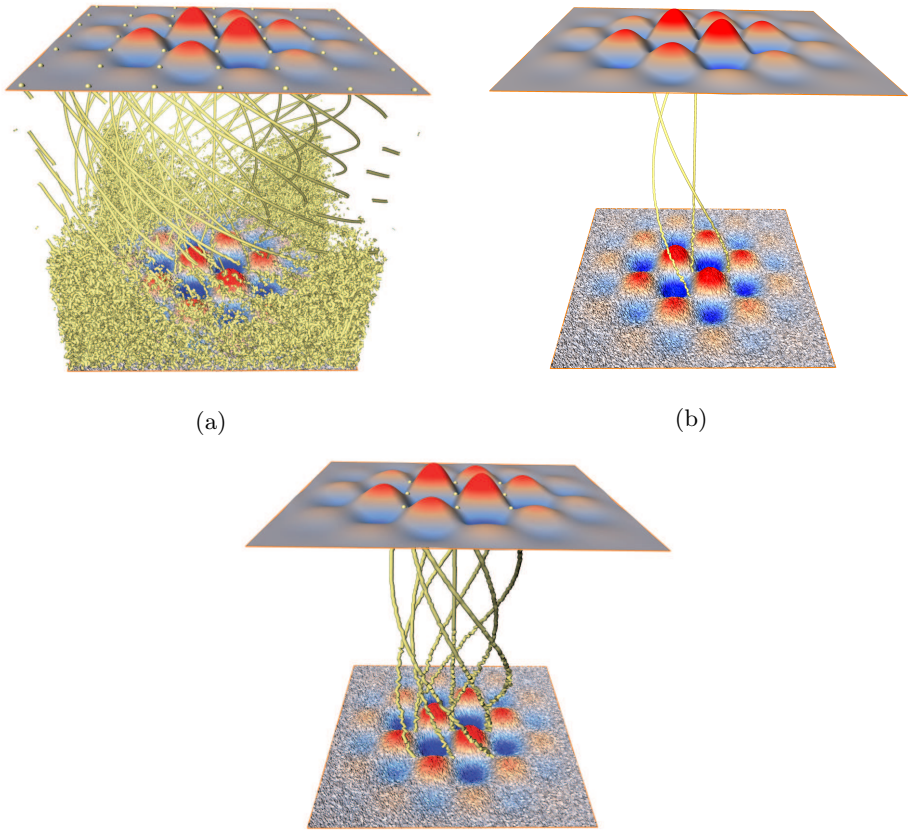
V_k (Lines 14-18). If this set of saddles contains u, we have found a critical saddle line in $V_{(k,k+1)}$ and append the pair $\{u, w\}$ to the result.

4.7.3 Evaluation

In this Section, we will evaluate the algorithm presented in Section 4.7.2. We show its robustness with respect to noise, compare it to the continuous Feature Flow Field tracking algorithm, and apply it to a real-world data set from computational fluid dynamics. We conclude the evaluation of our algorithm with a performance analysis.

Robustness

To demonstrate the ability of our algorithm to deal with noisy data we consider a synthetic data set. The data values are given by a 2D analytic function sampled on a uniform 256×256 mesh. A height field visualization of this function is shown in Figure 4.17. This data is then rotated to generate a sequence of 256 scalar fields (f_t) on the uniform mesh. To show the influence of the noise on the extraction methods we added an increasing amount of noise to the second half of the sequence (f_t).



(a) (b)

(c)

Figure 4.17 Evaluation of noise robustness on a synthetic data set – time is represented using the z-coordinate. a) critical saddle lines extracted with the Stable Feature Flow Field method. b) critical saddle lines of the Stable Feature Flow Field method filtered by line length. c) critical saddle lines extracted by our method using an appropriate persistence threshold σ.

We then applied the algorithm presented in Section 4.7.2 and the stabilized continuous Feature Flow Field method [WTGP10] to this data set. Figure 4.17 shows the critical saddle lines extracted by these two algorithms. Due to the

presence of noise, the continuous extraction method results in an overwhelming number of critical saddle lines. Note that some important lines are interrupted which implies that they are removed early when we filter the result by line length. In contrast, our combinatorial algorithm is able to extract all dominant critical saddle lines of this time-dependent data set using a persistence threshold σ slightly above the range of the noise.

Comparison

We compare our algorithm to the stabilized continuous Feature Flow Field method [WTGP10] using a data set from fluid dynamics [NSA+08]. The data set consists of a simulation of the time-dependent flow behind a cylinder. The data set is given on an adaptive mesh with $108k$ vertices and 320 time steps. We analyze the scalar quantity acceleration, a measure for vortex activity in fluid flows [KHNH10] depicted in Figure 4.19. For the combinatorial method we set the persistence threshold σ for the computation of the combinatorial gradient fields to about one percent of the data range.

Figure 4.18 shows the critical lines extracted by both methods in a small subregion of the data set. The continuous results are shown in the left (original data set) and middle column (a smoothed version of the data) while the combinatorial results are shown in the right column. The three rows show the critical minima lines (top), saddle lines (middle), and maxima lines (bottom).

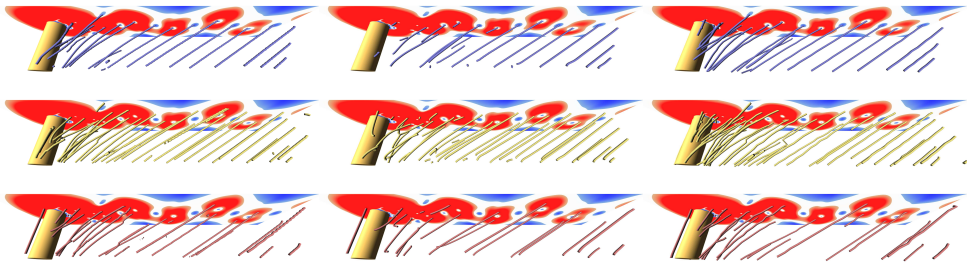


Figure 4.18 Comparison between the Stable Feature Flow Field method (left column) and our combinatorial method (right column) on a subset of the cylinder flow data set. The middle column depicts the result of the Stable Feature Flow Field method applied to a pre-smoothed version of the data set. Top row: critical minima lines. Middle row: critical saddle lines. Bottom row: critical maxima lines.

In general, both methods extract the correct critical lines in the right half of the depicted subregion of the data set. Some lines extracted by the continuous method do contain a very high oscillation. Applying the continuous method to the smoothed version of the data set removes these oscillations, but also removes some important critical lines.

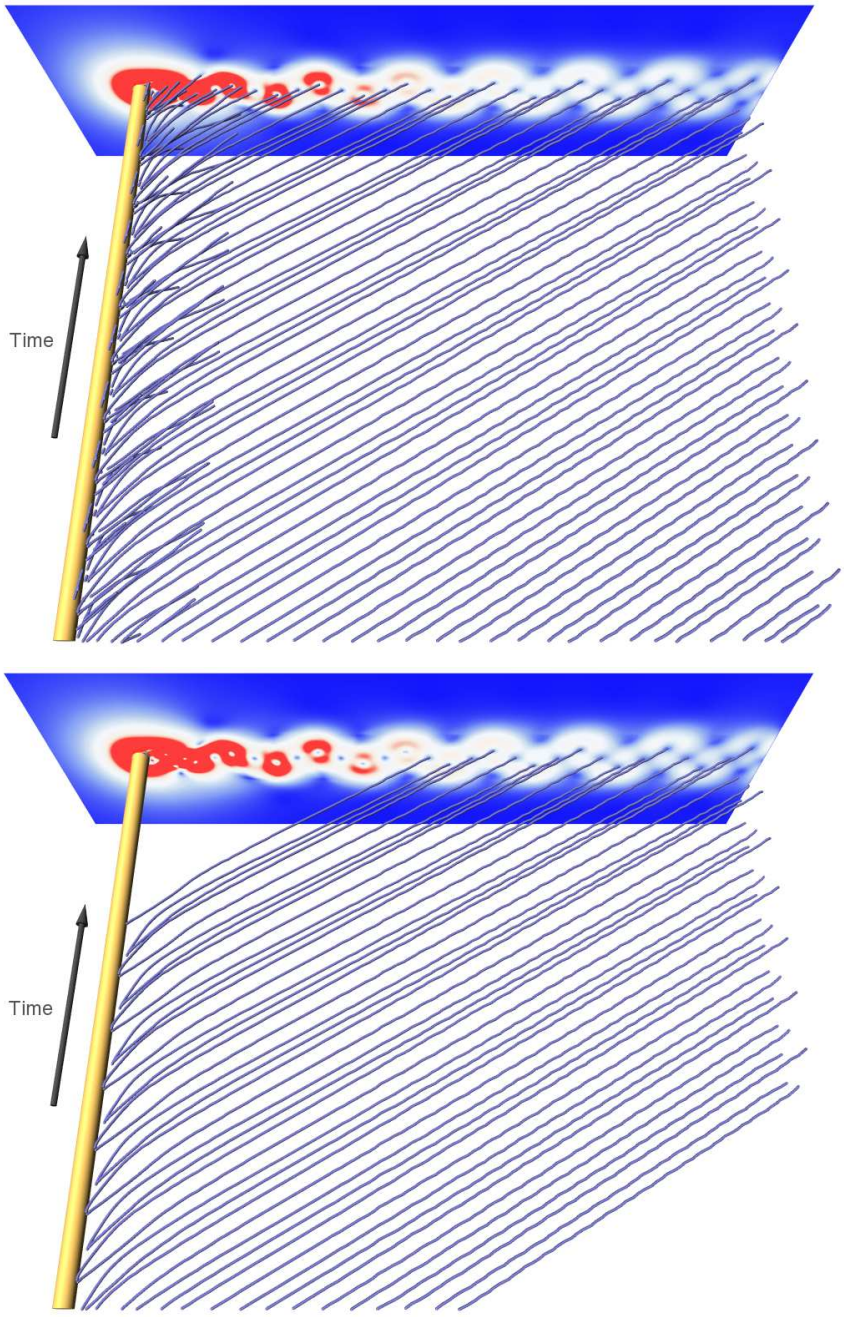


Figure 4.19 Evaluation of different filter criteria for critical minima lines (blue) of the acceleration of a flow dataset. The dominant minima of the acceleration describe vortex activity of the flow. Top: all extracted critical minima lines computed by Algorithm 9 without any post processing. Bottom: lines filtered by length. Continued in Figure 4.20

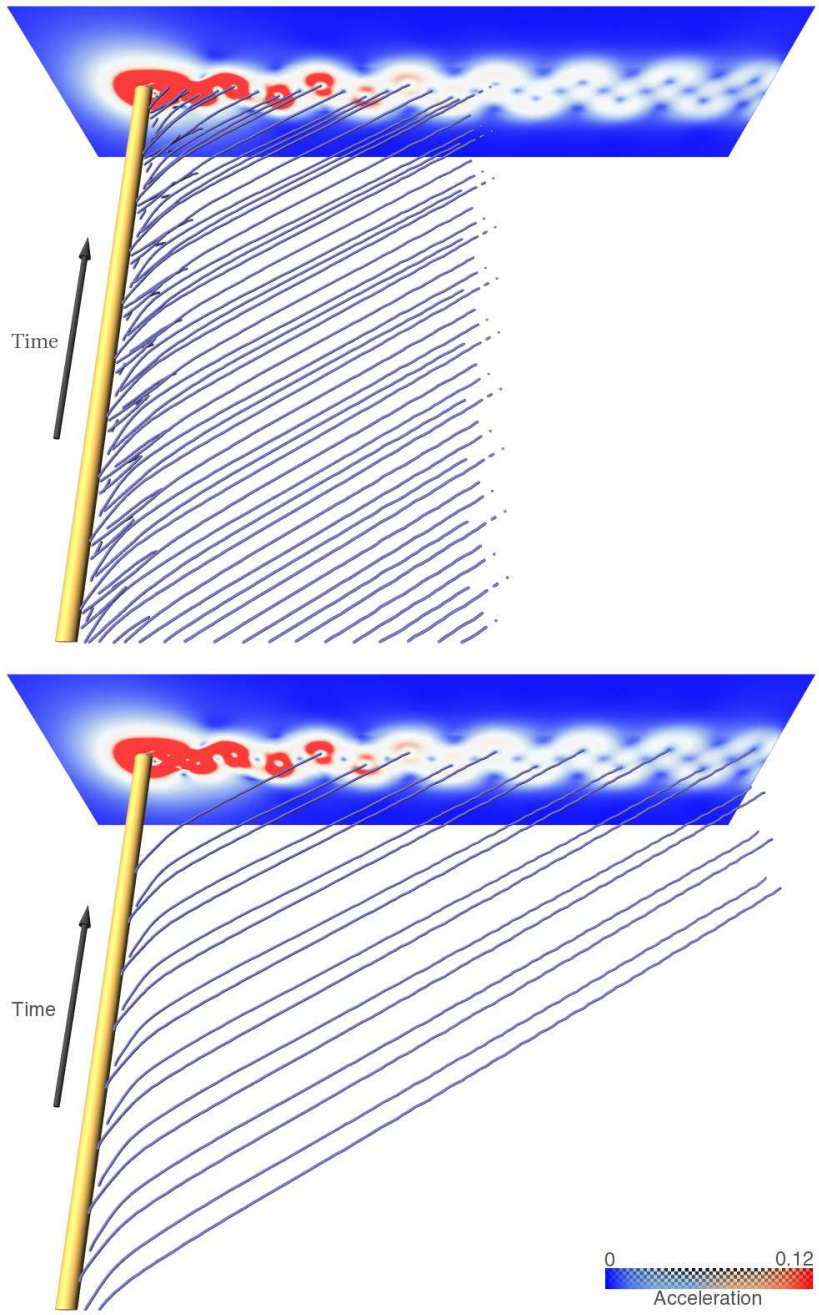


Figure 4.20 Continued from Figure 4.19. Top: lines filtered by spatial persistence. Bottom: lines filtered by our novel importance measure integrated persistence. The lines with high integrated persistence correspond to the dominant vortex activity of this data set as shown in [WSTH07].

Application

We applied our method to a scalar data set derived from a flow simulation [CSD03]. The simulation shows the flow over a cavity from left to right. Due to the cavity, there is a dominant vortex that separates from the wall after some time and moves through the cavity to the right side, where it hits the other wall.

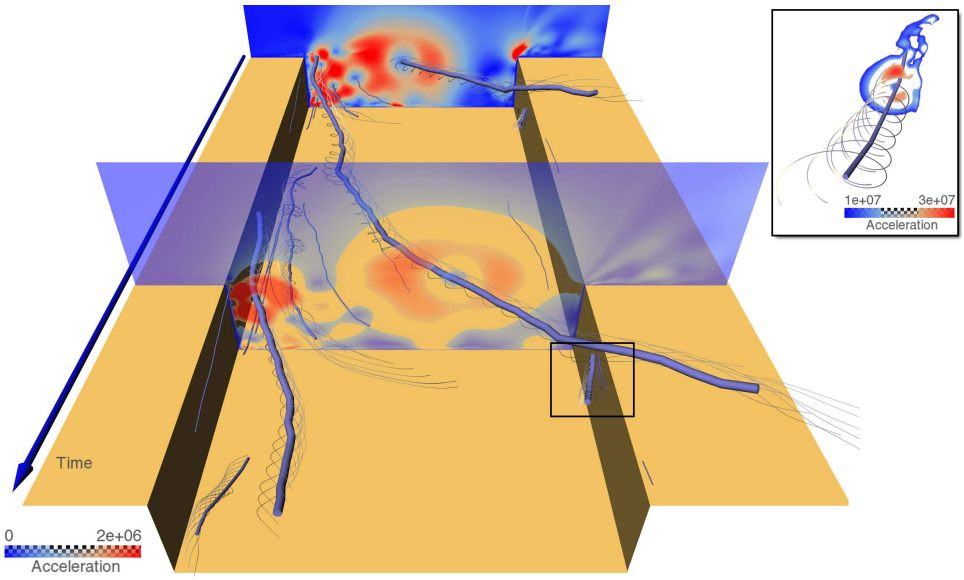


Figure 4.21 Application of our method to a real-world data set from computational fluid dynamics. The data set is the simulation of the flow over a cavity. The dominant minima of the acceleration of the flow describe the vortex activity. We extracted the critical minima lines of this data set using our method. The thickness of these lines is defined by our novel importance measure integrated persistence. To demonstrate the physical relevance of this importance measure, path lines are seeded in the vicinity of the most important lines. Note that there is a short, but important critical line in this data set (see close-up). This shows that the length of a critical line by itself is not a good importance measure in general.

As an indicator for time-dependent vortex structures, we used acceleration, a scalar quantity whose dominant minima indicate vortex activity [KHNH10, FKS+10]. Note that in contrast to the zeros of the velocity field, the minima of the acceleration do not depend on the chosen frame of reference. We computed the acceleration on the adaptive mesh that was used during the simulation of the flow consisting of $26k$ nodes for each of the 690 time steps. For the combinatorial computation of the critical lines, the persistence threshold σ was set to about one percent of the data range. Since we are only interested

in the minima of the acceleration, we only show the critical minimal lines in Figure 4.21. To demonstrate the physical significance of the importance measure introduced in Section 4.7.1, the thickness of the lines is determined by Integrated Persistence. The dominant vortices that pass through the cavity have a high Integrated Persistence. This can be visualized by seeding path lines in the vicinity of the lines with high Integrated Persistence.

Note that our algorithm has found one critical minima line that is difficult to observe manually (see zoom-in in Figure 4.21 and consider the color map therein). This short critical line has a higher Integrated Persistence than most other critical lines in the data set. By seeding path lines in its vicinity we observe that this line corresponds to strong vortex activity.

This example shows that it is in general problematic to use line length as an importance measure for critical lines.

Performance

The performance of our implementation was calculated for all three data sets used in this section. Table 4.3 contains the running times for a standard workstation containing two Intel Xeon E5620 CPUs. The table shows the number of data values given at the vertices of the grid and the number of slices T for which the critical lines were computed. Tracking the critical points in the computed CGFs is very fast - for a mesh with approximately one hundred thousand vertices, only 38 milliseconds are required for each time step.

Dataset	#vertices	T	CGF	Pers.	CFFF	Total
Synthetic	65k	256	77s	67s	5s	149s
Cavity	26k	690	186s	110s	4s	300s
Cylinder	108k	320	368s	96s	12s	476s

Table 4.3 Performance analysis of the CFFF method. For each data set shown in this section we measured the running time for the computation of the combinatorial gradient fields (CGF), the computation of the persistence values (Pers.), and the tracking of the critical points in the resulting sequence (CFFF).

For comparison, we have also measured the running time of the stable Feature Flow Field method. Computing the critical lines for the synthetic data set shown in Figure 4.17 with this method takes 333 seconds compared to the total running time of 149s using our method. Comparing the running time for the other data sets is problematic, since they are defined on an adaptive mesh and the implementation of the Feature Flow Field that is available to us can only be applied to uniform grids.

Note that the timings for the CGF computation represent the computation of the full hierarchy of CGFs (4.8). The user can thereby quickly select an appropriate persistence threshold σ in a post processing step.

4.7.4 Discussion

As shown in Section 4.7.3, our combinatorial algorithm to extract critical lines of time-dependent two dimensional scalar fields works very well in practice:

1. It effectively handles noisy data, see Figure 4.17.
2. It allows for a physically relevant importance measure for the tracked critical points, see Figures 4.19 and 4.21.
3. Its extracted features correspond to the results of the Feature Flow Field method for a smooth data set, see Figure 4.18.
4. It has a practical running time, see Table 4.3.

The robustness of our algorithm with respect to noise is mainly due to the notion of persistence which allows for a robust computation of a CGF. Unfortunately, using persistence can be problematic if the data contains outliers. To efficiently deal with such data, an importance measure for critical points would need to be developed that can address outliers in a sensible way.

Many of the existing tracking algorithms mentioned in Section 2.3 extract bifurcation points, i.e. the points where a pair of critical points appears or disappears. The spatial importance of such critical points becomes arbitrarily small as they approach a bifurcation point, see Figure 4.22.

Due to our focus on noise resilient extraction of critical lines, we do not aim at a precise computation of bifurcation points in this work. Note that critical points of course can appear or disappear in our method – we start tracking them as soon as their spatial importance is high enough to differentiate them from noise induced critical points.

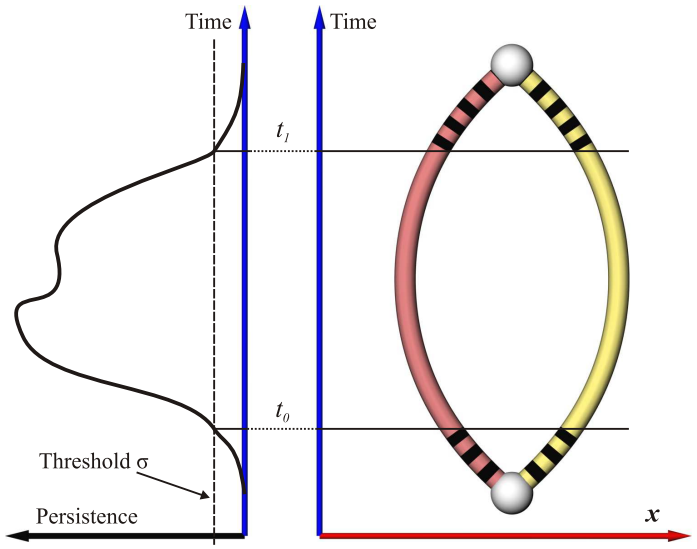


Figure 4.22 Bifurcation handling in CFFF. Right: a maximum-saddle pair evolving over time. Left: spatial importance of this pair over time. The pair is only tracked while the spatial importance is above the threshold σ. For $t < t_0$ and $t_1 < t$ the critical points are considered as noise.

Chapter 5

Conclusions and Outlook

The unified framework proposed in Chapter 4 has been shown to be well suited for two dimensional data of various types in Sections 4.4 - 4.7. We have shown that it is possible to treat vector fields, scalar fields, divergence-free vector fields, and time-dependent scalar fields using the generic pipeline described in Section 4.3.

This pipeline produces extremal structures that are always consistent with the topology of the domain of the given data set. This mathematical property is preserved in practice due to the combinatorial nature of the definitions and algorithms which allow for an exact binary representation. The topological consistence of the computed result significantly increases the robustness of our algorithm: a single critical point cannot be missed or misclassified, as this would make the resulting extremal structure topologically inconsistent.

This combinatorial nature is also a great advantage in practice, since no computational parameters have to be adjusted for each data set to get the desired result. This is an essential property for gaining insight through data analysis since it admits the objective discovery of new features in the data set. It is also beneficial for the automated analysis of a series of data sets since it is sometimes not practical to adjust numerical parameters for each element of the sequence. Also, the running time for a given data set size is quite predictable.

The concept of a hierarchy of extremal structures is also very useful in practice. It allows to treat noisy data sets directly and the user can choose to only extract the dominant features of the data set. In classical approaches, the data is often explicitly smoothed in a preprocessing step, or indirectly by robust gradient recovery schemes. This may disturb the true extremal structure of the data set – new critical points may be created, or existing ones may be destroyed. In contrast, our approach extracts all extremal structures and allows the user to remove noise by selecting an appropriate importance threshold, see e.g. Figure 4.10. It is also possible to restrict the output of our algorithm to the features with a high importance value. For example, the tiny

feature shown in the zoom-in in Figure 4.21 has a high importance, but would be quickly destroyed in a smoothing process due to its small spatial extent.

The assumption that the input data is defined on a cell complex is very general and allows to support most mesh types encountered in practice using a well defined interface. The clear separation of the mesh type from the data type allows complex algorithms [GRP$^+$] to be applicable to all mesh types without any modifications. It also allows to exploit regular structures of the mesh transparently. For example, in [GRWH] the regular structure of lattices was heavily exploited to propose a memory efficient algorithm for 3D image data. While such an implicit representation of a cubical cell complex is quite involved, it only has to be written once and all algorithms contained in this thesis can profit from it.

There are two main areas where the existing framework can be improved.

The first area is the extension and evaluation of the existing algorithms to higher dimensions:

- While the algorithm for vector fields proposed in Section 4.4 is applicable to any dimension, it is not known how the practical complexity and the approximation quality of the algorithm would be affected by the different structure of a high dimensional cell complex. Also, the efficient computation of the k-orbits with $0 < k < d - 1$ is an open problem.

- For scalar valued data, an extension of the algorithm described in Section 4.5 to three dimension has been proposed and evaluated in [GRP$^+$]. An algorithm with mathematical guarantees is also available [RWS11]. In contrast, the treatment of even higher dimensional data sets is unclear and could even be unfeasible due to topological obstructions, see [Bau11, RWS11].

- The current approach to divergence-free vector fields proposed in Section 4.6 is restricted to two dimensions and it is not clear how one would extent it to higher dimensions. A good starting point for a combinatorial treatment may consist of combinatorial Novikov-Morse theory [For02].

- The combinatorial feature flow field definition introduced in Section 4.7 for time-dependent scalar fields can be directly extended to higher dimensions. The main algorithmic challenge is the efficient extraction of the critical lines. For two dimensions this is shown in Section 4.7.2. It is not known whether the same ideas can be applied to track saddles in higher dimensions.

  Tracking minima and maxima in higher dimension however is doable – a close inspection of the definition of combinatorial critical minima and maxima lines in higher dimensions reveals that they have the same combinatorial structure as in two dimension. Given an algorithm that

can compute a combinatorial gradient field for a high dimensional data set we can therefore directly use our algorithm to track its minima and maxima.

The second area concerns the geometric embedding of the separatrices and orbits. As can be seen in Figure 4.7, the combinatorial streamlines do not follow the streamlines of the continuous vector field. In fact, a periodic orbit with the shape of a circle will always be approximated by an octagon on a uniform mesh in the combinatorial setting – regardless of the mesh resolution. While the geometric embedding of the separatrices is fine for many data sets, see e.g. Figure 4.11, it would be beneficial to have an algorithm whose combinatorial separatrices and orbits converge to the analytic ones with increasing mesh resolution.

Looking for such an algorithm that is still combinatorial, topologically consistent, and parameter-free, is probably a difficult but highly rewarding endeavor.

Bibliography

[Aim] AimAtShape, http://shapes.aim-at-shape.net/.

[Bau11] Ulrich Bauer, *Persistence in discrete morse theory*, Ph.D. thesis, University of Göttingen, 2011.

[Bel58] Richard Bellman, *On a routing problem*, Quarterly of Applied Mathematics **16** (1958), no. 1, 87–90.

[BLW10] Ulrich Bauer, Carsten Lange, and Max Wardetzky, *Optimal topological simplification of discrete functions on surfaces*, arXiv:1001.1269v2 (2010).

[BP02] Dirk Bauer and Ronald Peikert, *Vortex tracking in scale space*, Joint Eurographics — IEEE TCVG Symposium on Visualization, May 2002, pp. 140–147.

[BSS02] Chandrajit Bajaj, Ariel Shamir, and Bong-Soo Sohn, *Progressive tracking of isosurfaces in time-varying scalar fields*, Tech. Report TR-02-4, CS & TICAM, Department of Computer Sciences & TICAM, University of Texas Austin, 2002.

[BWP+10] Peer-Timo Bremer, Gunther H. Weber, Valerio Pascucci, Marc Day, and John B. Bell, *Analyzing and tracking burning structures in lean premixed hydrogen flames*, IEEE Transactions on Visualization and Computer Graphics **16** (2010), no. 2, 248–260.

[Cay59] A. Cayley, *On contour and slope lines*, The London, Edinburg and Dublin Philosophical Magazine and Journal of Science **18** (1859), 264–268.

[CCA+05] J.R. Cebral, M.A. Castro, S. Appanaboyina, C.M. Putman, D. Millan, and A.F. Frangi, *Efficient pipeline for image-based patient-specific analysis of cerebral aneurysm hemodynamics: technique and sensitivity*, IEEE transactions on medical imaging **24** (2005), no. 4, 457–467.

[Cha00] Manoj K. Chari, *On discrete morse functions and combinatorial decompositions*, Discrete Math. **217** (2000), no. 1-3, 101–113.

[CJR07] Jesus Caban, Alark Joshi, and Penny Rheingans, *Texture-based feature tracking for effective time-varying data visualizations*, IEEE Transactions on Visualization and Computer Graphics (Vis07) **13** (2007), no. 6, 1472–1479.

[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 in Visualization and Computer Graphics **13** (2007), 769–785.

[CMLZ08] Guoning Chen, Konstantin Mischaikow, Robert S. Laramee, and Eugene Zhang, *Efficient morse decomposition of vector fields*, IEEE Transaction on Visualization and Computer Graphics **14** (2008), no. 4, 848–862.

[Con78] C. Conley, *Isolated invariant sets and the morse index*, AMS Conference Board of the Mathematical Sciences Series No. 38 (1978).

[CSD03] Edgar Caraballo, Mo Samimy, and J. DeBonis, *Low dimensional modeling of flow for closed-loop flow control*, AIAA Paper **59** (2003).

[CSEM06] David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov, *Vines and vineyards by updating persistence in linear time*, Proceedings of the twenty-second annual symposium on Computational geometry (New York, NY, USA), SCG '06, ACM, 2006, pp. 119–126.

[Die97] R. Diestel, *Graph theory*, Springer-Verlag, Heidelberg, Germany, 1997.

[Dij59] Edsger. W. Dijkstra, *A note on two problems in connexion with graphs.*, Numerische Mathematik **1** (1959), 269–271.

[DLL+10] Tamal K. Dey, K. Li, Chuanjiang Luo, Pawas Ranjan, Issam Safa, and Yusu Wang, *Persistent heat signature for pose-oblivious matching of incomplete models*, Comput. Graph. Forum **29** (2010), no. 5, 1545–1554.

[dLvL01] Wim de Leeuw and Robert van Lieres, *Chromatin decondensation: A case study of tracking features in confocal data*, VIS '01: Proceedings of the conference on Visualization '01 (Washington, DC, USA), IEEE Computer Society, 2001, pp. 441–444.

[EH04] Herbert Edelsbrunner and J. Harer, *Jacobi sets of multiple Morse functions*, Foundations of Computational Mathematics, Minneapolis 2002 (F. Cucker, R. DeVore, P. Olver, and E. Sueli, eds.), Cambridge Universtiy Press, 2004, pp. 37–57.

[EH08] ———, *Persistent homology — a survey*, Surveys on Discrete and Computational Geometry: Twenty Years Later (J. E. Goodman, J. Pach, and R. Pollack, eds.), vol. 458, AMS Bookstore, 2008, pp. 257–282.

[EHM+08] Herbert Edelsbrunner, John Harer, Ajith Mascarenhas, J. Snoeyink, and Valerio Pascucci, *Time-varying Reeb graphs for continuous space-time data*, Computation Geometry: Theory and Applications **41** (2008), no. 3, 149–166.

[EHNP03] Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci, *Morse-smale complexes for piecewise linear 3-manifolds*, SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry (New York, NY, USA), ACM, 2003, pp. 361–370.

[ELZ02] H. Edelsbrunner, D. Letscher, and A. Zomorodian, *Topological persistence and simplification*, Discrete Computational Geometry **28** (2002), 511–533.

[FKS+10] Raphael Fuchs, Jan Kemmler, Benjamin Schindler, Filip Sadlo, Helwig Hauser, and Ronald Peikert, *Toward a Lagrangian Vector Field Topology*, Computer Graphics Forum **29** (2010), no. 3, 1163–1172.

[For56] L.R. Ford, *Network flow theory*, Paper P-923, The RAND Corperation, Santa Moncia, California, August 1956.

[For98a] Robin Forman, *Combinatorial vector fields and dynamical systems*, Mathematische Zeitschrift **228** (1998), 629–681.

[For98b] Robin Forman, *Morse theory for cell complexes*, Advances in Mathematics **134** (1998), 90–145.

[For01] Robin Forman, *A user's guide to discrete morse theory*, Proceedings of the 2001 Internat. Conf. on Formal Power Series and Algebraic Combinatorics, Advances in Applied Mathematics, 2001.

[For02] ———, *Combinatorial novikov-morse theory*, International Journal of Mathematics **13** (2002), 333–368.

[FT87] Michael L. Fredman and Robert Endre Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, J. ACM **34** (1987), 596–615.

[GBHP08] Attila Gyulassy, Peer-Timo Bremer, Bernd Hamann, and Valerio Pascucci, *A practical approach to Morse-Smale complex computation: scalability and generality*, IEEE Transactions on Visualization and Computer Graphics **14** (2008), 1619–1626.

[GBPH11] Attila Gyulassy, Peer-Timo Bremer, Valerio Pascucci, and Bernd Hamann, *Practical considerations in Morse-Smale complex computation*, Topological Methods in Data Analysis and Visualization (Valerio Pascucci, Xavier Tricoche, Hans Hagen, and Julien Tierny, eds.), Mathematics and Visualization, Springer Berlin Heidelberg, 2011, pp. 67–78.

[GNP+06] Attila Gyulassy, Vijay Natarajan, Valerio Pascucci, Peer-Timo Bremer, and Bernd Hamann, *A topological approach to simplification of three-dimensional scalar functions*, IEEE Transactions on Visualization and Computer Graphics **12** (2006), no. 4, 474–484.

[GNPH07] Attila Gyulassy, Vijay Natarajan, Valerio Pascucci, and Bernd Hamann, *Efficient computation of Morse-Smale complexes for three-dimensional scalar functions*, IEEE Transactions on Visualization and Computer Graphics **13** (2007), 1440–1447.

[GRP+] David Günther, Jan Reininghaus, Steffen Prohaska, Tino Weinkauf, and Hans-Christian Hege, *Efficient computation of a hierarchy of discrete 3d gradient vector fields*, Topological Methods in Data Analysis and Visualization II (Ronald Peikert, Hamish Carr, and Helwig Hauser, eds.), Mathematics and Visualization, Springer Berlin Heidelberg, to appear.

[GRWH] David Günther, Jan Reininghaus, Hubert Wagner, and Ingrid Hotz, *Memory-efficient computation of persistent homology for 3d images using discrete morse theory*, Proceedings Sibgrapi 2011 – Conference on Graphics, Patterns and Images, to appear.

[GTS04] Christoph Garth, Xavier Tricoche, and Gerik Scheuermann, *Tracking of vector field singularities in unstructured 3d time-dependent datasets*, VIS '04: Proceedings of the conference on Visualization '04 (Washington, DC, USA), IEEE Computer Society, 2004, pp. 329–336.

[Gyu08] Attila Gyulassy, *Combinatorial construction of morse-smale complexes for data analysis and visualization.*, Ph.D. thesis, University of California, Davis, 2008.

[Hat02] A. Hatcher, *Algebraic topology*, Cambridge University Press, Cambridge, U.K., 2002.

[HD04] S. Hougardy and D.E. Drake, *Approximation algorithms for the weighted matching problem*, Oberwolfach Report 28, 2004.

[Hel58] Hermann Helmholtz, *Über integrale der hydrodynamischen gleichungen, welche den wirbelbewegungen entsprechen*, J. Reine Angew. Math. **55** (1858), 25–55.

[HH89] J. Helman and L. Hesselink, *Representation and display of vector field topology in fluid flow data sets*, Computer **22** (1989), no. 8, 27–36.

[HN07] P. Harish and P.J. Narayanan, *Accelerating large graph algorithms on the gpu using cuda*, Proc of IEEE International Conference on High Performance Computing, 2007.

[Ji06] Guangfeng Ji, *Feature tracking and viewing for time-varying data sets*, Ph.D. thesis, Ohio State University, Columbus, OH, USA, 2006, Adviser-Shen, Han-Wei.

[JP06] Michael Joswig and Marc E. Pfetsch, *Computing optimal Morse matchings*, SIAM J. Discret. Math. **20** (2006), no. 1, 11–25.

[JSW03] Guangfeng Ji, Han-Wei Shen, and Rephael Wenger, *Volume tracking using higher dimensional isosurfacing*, VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03) (Washington, DC, USA), IEEE Computer Society, 2003, pp. 209–216.

[KE07] Thomas Klein and Thomas Ertl, *Scale-space tracking of critical points in 3d vector fields*, Topology-based Methods in Visualization (Hans Hagen Helwig Hauser and Holger Theisel, eds.), Mathematics and Visualization, Springer Berlin Heidelberg, May 2007, pp. 35–49.

[KHNH10] J. Kasten, I. Hotz, B.R. Noack, and H.-C. Hege, *On the extraction of long-living features in unsteady fluid flows*, Topological Methods in Data Analysis and Visualization. Theory, Algorithms, and Applications. (Valerio Pascucci, Xavier Tricoche, Hans Hagen, and Julien Tierny, eds.), Springer, 2010, pp. 115–126.

[KKM05] Henry King, Kevin Knudson, and Neza Mramor, *Generating discrete Morse functions from point data*, Experimental Mathematics **14** (2005), no. 4, 435–444.

[KKM08] _____, *Birth and death in discrete morse theory*, arXiv:0808.0051v1 (2008).

[KRHH11] Jens Kasten, Jan Reininghaus, Ingrid Hotz, and Hans-Christian Hege, *Two-dimensional time-dependent vortex regions based on the acceleration magnitude.*, IEEE Trans. Vis. Comput. Graph. **17** (2011), no. 12, 2080–2087.

[Kuh55] H. Kuhn, *The hungarian method for the assignment problem*, Naval Research Logistics Quarterly **2** (1955), 83–97.

[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 **12** (2006), no. 5, 1053–1060.

[Lew02] Thomas Lewiner, *Constructing discrete morse function*, Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, Department of Mathematics, 2002.

[Lew05] ———, *Geometric discrete Morse complexes*, Ph.D. thesis, Department of Mathematics, PUC-Rio, 2005, Advised by Hélio Lopes and Geovan Tavares.

[LHZP07] Robert S. Laramee, Helwig Hauser, Lingxiao Zhao, and Frits H. Post, *Topology-based flow visualization, the state of the art*, Topology-based Methods in Visualization (Hans Hagen Helwig Hauser and Holger Theisel, eds.), Mathematics and Visualization, Springer Berlin Heidelberg, May 2007, pp. 1–19.

[LLT04] Thomas Lewiner, Helio Lopes, and Geovan Tavares, *Applications of forman's discrete morse theory to topology visualization and mesh compression*, IEEE Transactions on Visualization and Computer Graphics **10** (2004), no. 5, 499–508.

[Max70] J. C. Maxwell, *On hills and dales*, The London, Edinburg and Dublin Philosophical Magazine and Journal of Science **40** (1870), 421–425.

[Mil63] John Milnor, *Morse theory*, Princeton University Press, 1963.

[Mil65] J.W. Milnor, *Topology from the differentiable viewpoint*, Univ. Press Virginia, 1965.

[MKFI97] X. Mao, M. Kikukawa, N. Fujeta, and N. Imamiya, *Line integral convolution for 3d surfaces*, Visualization in Scientific Computing '97 (W. Lefer and M. Grave, eds.), Springer, Berlin, 1997, pp. 57–69.

[Mun84] J. R. Munkres, *Elements of algebraic topology*, Addison-Wesley, Redwood City, 1984.

[NSA+08] B. R. Noack, M. Schlegel, B. Ahlborn, G. Mutschke, M. Morzyński, P. Comte, and G. Tadmor, *A finite-time thermodynamics of unsteady fluid flows*, J. Non-Equilibr. Thermodyn. **33** (2008), no. 2, 103–148.

[Pan84] R.W. Panton, *Incompressible Flow*, John Wiley & Sons, New York, etc., 1984.

[Pos03] Frits H. Post, *The state of the art in flow visualization: Feature extraction and tracking*, Computer Graphics Forum (David Duke and Roberto Scopigno, eds.), vol. 22(4), Blackwell Publishing Inc, Oxford, UK and Boston, USA, June 2003, pp. 775–792.

[PP02] Konrad Polthier and Eike Preuß, *Identifying vector field singularities using a discrete Hodge decomposition*, Springer Verlag, 2002, pp. 112–134.

[PPL+10] Fabiano Petronetto, Afonso Paiva, Marcos Lage, Geovan Tavares, Hélio Lopes, and Thomas Lewiner, *Meshless Helmholtz-Hodge decomposition*, Transactions on Visualization and Computer Graphics **16** (2010), no. 2, 338–342.

[RGH+10] Jan Reininghaus, David Günther, Ingrid Hotz, Steffen Prohaska, and Hans-Christian Hege, *TADD: A computational framework for data analysis using discrete Morse theory*, Mathematical Software – ICMS 2010 (Komei Fukuda, Joris van der Hoeven, Michael Joswig, and Nobuki Takayama, eds.), Lecture Notes in Computer Science, vol. 6327, Springer, 2010, pp. 198–208.

[RH] Jan Reininghaus and Ingrid Hotz, *Computational discrete morse theory for divergence-free 2d vector fields*, Topological Methods in Data Analysis and Visualization II (Ronald Peikert, Hamish Carr, and Helwig Hauser, eds.), Mathematics and Visualization, Springer Berlin Heidelberg, to appear.

[RH11] Jan Reininghaus and Ingrid Hotz, *Combinatorial 2d vector field topology extraction and simplification*, Topological Methods in Data Analysis and Visualization (Valerio Pascucci, Xavier Tricoche, Hans Hagen, and Julien Tierny, eds.), Mathematics and Visualization, Springer Berlin Heidelberg, 2011, pp. 103–114.

[RKG+11] Jan Reininghaus, Natallia Kotava, David Guenther, Jens Kasten, Hans Hagen, and Ingrid Hotz, *A scale space based persistence measure for critical points in 2d scalar fields*, IEEE Transactions on Visualization and Computer Graphics **17** (2011), 2045–2052.

[RKWH11] Jan Reininghaus, Jens Kasten, Tino Weinkauf, and Ingrid Hotz, *Efficient computation of combinatorial feature flow fields*, IEEE Transactions on Visualization and Computer Graphics **99** (2011), no. PrePrints.

[RLH11] Jan Reininghaus, Christian Löwen, and Ingrid Hotz, *Fast combinatorial vector field topology*, IEEE Transactions on Visualization and Computer Graphics **17** (2011), 1433–1443.

[RPP+09] Olufemi Rosanwo, Christoph Petz, Steffen Prohaska, Ingrid Hotz, and Hans-Christian Hege, *Dual streamline seeding*, Proceedings of the IEEE Pacific Visualization Symposium (Peter Eades, Thomas Ertl, and Han-Wei Shen, eds.), 2009, pp. 9 – 16.

[RPS99] F. Reinders, F.H. Post, and H.J.W. Spoelder, *Attribute-based feature tracking*, Proceedings of EG - IEEE TCVG Symposium on Visualization '99, 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** (2002), no. 4, 675–682.

[RWS11] Vanessa Robins, Peter John Wood, and Adrian P. Sheppard, *Theory and algorithms for constructing discrete morse complexes from grayscale digital images.*, IEEE Trans. Pattern Anal. Mach. Intell. **33** (2011), no. 8, 1646–1658.

[SB06] Bong-Soo Sohn and Chandrajit Bajaj, *Time-varying contour topology*, IEEE Transactions on Visualization and Computer Graphics **12** (2006), no. 1, 14–25.

[Sch03] Alexander Schrijver, *Combinatorial optimization*, Springer, 2003.

[SG08] J.J. Sanchez-Gabites, *Dynamical systems and shapes*, RACSAM: Geometry and Topology **102** (2008), 127–159.

[SH95] D. Stalling and H. Hege, *Fast and resolution independent line integral convolution*, Proceedings Siggraph '95 (1995), 249–256, Los Angeles.

[Sho09] Clayton Shonkwiler, *Poincaré duality angles for Riemannian manifolds with boundary*, Tech. Report arXiv:0909.1967, Sep 2009, Comments: 51 pages, 6 figures.

[Sma61] Stephen Smale, *On gradient dynamical systems*, The Annals of Mathematics **74** (1961), 199–206.

[SSZC94] Ravi Samtaney, Deborah Silver, Norman Zabusky, and Jim Cao, *Visualizing features and tracking their evolution*, Computer **27** (1994), no. 7, 20–27.

[SW97] Deborah Silver and Xin Wang, *Tracking and visualizing turbulent 3d features*, IEEE Transactions on Visualization and Computer Graphics **3** (1997), no. 2, 129–141.

[SWH05] D. Stalling, M. Westerhoff, and H.-C. Hege, *Amira: A highly interactive system for visual data analysis*, The Visualization Handbook (2005), 749–767.

[TLHD03] Yiying Tong, Santiago Lombeyda, Anil N. Hirani, and Mathieu Desbrun, *Discrete multiscale vector field decomposition*, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH, vol. 22, 2003.

[TS03] Holger Theisel and Hans-Peter Seidel, *Feature flow fields*, VisSym '03: Proceedings of the symposium on Data Visualization 2003 (Aire-la-Ville, Switzerland, Switzerland), Eurographics Association, 2003, pp. 141–148.

[TSH01] Xavier Tricoche, Gerik Scheuermann, and Hans Hagen, *Continuous topology simplification of planar vector fields*, VIS '01: Proceedings of the conference on Visualization '01 (Washington, DC, USA), IEEE Computer Society, 2001, pp. 159–166.

[TSHC01] Xavier Tricoche, Gerik Scheuermann, Hans Hagen, and Stefan Clauss, *Vector and tensor field topology simplification on irregular grids*, VisSym '01: Proceedings of the symposium on Data Visualization 2001 (Wien, Austria) (D. Ebert, J. M. Favre, and R. Peikert, eds.), Springer-Verlag, May 28–30 2001, pp. 107–116.

[TWHS04] Holger Theisel, Tino Weinkauf, Hans-Christian Hege, and Hans-Peter Seidel, *Grid-independent detection of closed stream lines in 2d vector fields*, Proceedings of the VMV Conference 2004 (Stanford, USA), November 2004, p. 665.

[TWSH02] Xavier Tricoche, Thomas Wischgoll, Gerik Scheuermann, and Hans Hagen, *Topology tracking for the visualization of time-dependent two-dimensional flows*, Computer & Graphics **26** (2002), 249–257.

[WB98] Chris Weigle and David C. Banks, *Extracting iso-valued features in 4-dimensional scalar fields*, VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization (New York, NY, USA), ACM, 1998, pp. 103–110.

[WBD+ar] Gunther Weber, Peer-Timo Bremer, Marcus S. Day, John B. Bell, and Valerio Pascucci, *Feature tracking using reeb graphs*, TopoInVis'09, 2010 to appear.

[Wei08] T. Weinkauf, *Extraction of topological structures in 2d and 3d vector fields*, Ph.D. thesis, University Magdeburg and Zuse Institute Berlin, 2008.

[WG09] T. Weinkauf and D. Günther, *Separatrix persistence: Extraction of salient edges on surfaces using topological methods*, Computer Graphics Forum (Proc. SGP '09) **28** (2009), no. 5, 1519–1528.

[WS01] Thomas Wischgoll and Gerik Scheuermann, *Detection and visualization of closed streamlines in planar flows*, IEEE Transactions on Visualization and Computer Graphics **7** (2001), no. 2, 165–172.

[WST+07] T. Weinkauf, J. Sahner, H. Theisel, H.-C. Hege, and H.-P. Seidel, *A unified feature extraction architecture*, Active Flow Control (R. King, ed.), Notes on Numerical Fluid Mechanics and Multidisciplinary Design (NNFM), Springer, 2007, Active Flow Control 2006, Berlin, Germany, September 27 - 29, pp. 119–133.

[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** (2007), no. 6, 1759–1766.

[WTGP10] T. Weinkauf, H. Theisel, A. Van Gelder, and A. Pang, *Stable feature flow fields*, IEEE Transactions on Visualization and Computer Graphics (2010), accepted.

[WTS+05] Tino Weinkauf, Holger Theisel, K. Shi, Hans-Christian Hege, and Hans-Peter Seidel, *Extracting higher order critical points and topological simplification of 3D vector fields*, Proc. IEEE Visualization 2005 (Minneapolis, U.S.A.), October 2005, pp. 559–566.

[YJS06] Alper Yilmaz, Omar Javed, and Mubarak Shah, *Object tracking: A survey*, ACM Comput. Surv. **38** (2006), no. 4, 13.

List of Figures

1.1 Two algorithmic challenges 2

3.1 Graphs 10
3.2 Paths and cycles 11
3.3 Breadth first search algorithm 11
3.4 Bellman-Ford algorithm 13
3.5 Bipartite matchings 13
3.6 Augmenting path 14
3.7 Hungarian method 16
3.8 Surfaces 18
3.9 Simplices 20
3.10 Simplicial complex 20
3.11 Boundary operator 21
3.12 Homology 22
3.13 Morse theory 25
3.14 Morse-Smale complex 26
3.15 Discrete Morse functions 27
3.16 Combinatorial gradient field 28
3.17 Separatrices 28
3.18 Periodic orbits 30
3.19 Manifold-like simplicial complex 31

4.1 Cell graph 34
4.2 Basic definitions of graph theoretic discrete Morse theory 35
4.3 Extremal structures of noisy scalar data 37
4.4 Illustration of the hierarchical representation of the extremal structure using augmenting paths 38
4.5 Comparison of algorithms for vector fields 45
4.6 Approximation vs. exact vector field method 50
4.7 Vector field from biofluid mechanics 52
4.8 Importance measure illustration for vector field data 53
4.9 Comparison with continuous vector field topology 54
4.10 Synthetic noisy scalar field 60
4.11 Extremal lines in curvature fields 62

4.12 A synthetic divergence-free vector field is depicted using a LIC image colored by magnitude 67
4.13 Divergence-free vector field application 68
4.14 Combinatorial feature flow fields 72
4.15 Computational pipeline of CFFF 73
4.16 Illustration of saddle tracking in CFFF 75
4.17 Evaluation of noise robustness in CFFF 79
4.18 Comparison of the Stable Feature Flow Field method and CFFF . 80
4.19 Evaluation of different filter criteria in CFFF, part 1 81
4.20 Evaluation of different filter criteria in CFFF, part 2 82
4.21 Application of CFFF 83
4.22 Bifurcation handling in CFFF 86

List of Tables

4.1 Running time of the vector field algorithm 55
4.2 Running time of the scalar field algorithm 61
4.3 Running time of the time-dependent scalar field algorithm 84

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.

Jan Reininghaus
2. Dezember 2011